NEGOTIATING SOFTWARE

Redistributing Control at Work and on the Web

MIDAS NOUWENS

PhD dissertation
Department of Digital Design & Information Studies
Communication & Culture
Aarhus University

March 2021

Negotiating Software: Redistributing Control at Work and on the Web
by Midas Nouwens

A dissertation submitted in partial fulfilment of the requirements of Aarhus University for the degree of Doctor of Philosophy.

MAIN SUPERVISOR:
Clemens Nylandsted Klokmose

CO-SUPERVISOR:
Peter Dalsgaard

DOI: 10.7146/aulsps-e.480
ISBN: 978-87-7507-539-3

All that is gold does not glitter,
Not all those who wander are lost;
The old that is strong does not wither,
Deep roots are not reached by the frost.

From the ashes a fire shall be woken,
A light from the shadows shall spring;
Renewed shall be blade that was broken,
The crownless again shall be king.

— J. R. R. Tolkien

We live in capitalism. Its power seems inescapable.

So did the divine right of kings.

— Ursula K. Le Guin

ABSTRACT

Since the 1970s, digital technologies increasingly determine who gets what, when, and how; and the workings of informational capitalism have concentrated control over those technologies into the hands of just a few private corporations. The normative stance of this dissertation is that control over software should be distributed and subject to processes of negotiation: consensus-based decision making oriented towards achieving collective benefits. It explores the boundaries of *negotiating software* by trying to effect a change in two different kinds of software using two different approaches.

The first approach targets application software – the paradigmatic model of commodified, turn-key computational media – in the context of knowledge work – labour that involves the creation and distribution of information through non-routine, creative, and abstract thinking. It tries to effect change by developing *negotiable software* as an alternative to the autocratic application model, which is software that embeds the support for distributed control in and over its design. These systems successfully demonstrate the technological feasibility of this approach, but also the limitations of design as a solution to systemic power asymmetries.

The second approach targets consent management platforms – pop-up interfaces on the web that capture visitor's consent for data processing – in the context of the European Union's data protection regulation. It tries to effect change by employing *negotiation software*, which is software that supports existing processes of negotiation in complex systems, i.e., regulatory oversight and the exercise of digital rights. This approach resulted in a considerable increase in data protection compliance on Danish websites, but showed that sustainable enforcement using digital tools also requires design changes to data processing technologies.

Both approaches to effecting software change – making software negotiable and using software in negotiations – revealed the drawbacks of individualistic strategies. Ultimately, the capacity of the liberal subject to stand up against corporate power is limited, and more collective approaches to software negotiation need to be developed, whether when making changes to designs or leveraging regulation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Setting the Agenda

1.

INTRODUCTION

This dissertation is about how to negotiate software – the process of making a substantial and sustainable change to a software's design. It explores this by taking two types of software – workplace applications and web-consent pop-ups – and trying to renegotiate their designs.

Why should we care about negotiating software? Because software plays an important role in global patterns of power, and because control over that software is distributed unjustly.

1.1 SOFTWARE POWER

In the last half century, the political economies of most OECD countries have been transforming from *industrial* capitalism to *informational* capitalism. This qualifying adjective to capitalism follows Castell's seminal "The Rise of the Network Society",[1] in which he augments the Marxist concept of a society's *mode of production* (capitalism, feudalism) with the idea of a *mode of development* (industrialism, informationalism). A mode of production refers to the systematic way that surplus value is generated and controlled. The capitalist mode of production "is oriented toward ... increasing the amount of surplus appropriated by capital on the basis of the private control over the means of production and circulation".[2] Briefly explained, to create a surplus workers have been separated from the tools and resources they need to produce goods and services. This means that workers have to enter into a relation with the owners of those means of production to subsist (i.e., they exchange their labour for a wage). The owners of the means of production – the holders of capital – sell the goods and services produced by the workers for more than the costs needed to produce them, which means they can appropriate the surplus value. This framework of a mode of production helps explain how we structure our societies, who we interact with, what determines our quality of life, how power is distributed.

Castell introduces the idea of a *mode of development* to explain differences in capitalist societies across time and space. The mode of development tries to ex-

[1] Manuel Castells (2009). *The Rise of the Network Society*. 2nd ed. Vol. 1. The Information Age: Economy, Society, and Culture. Blackwell Publishers. ISBN: 978-0-631-22140-1.

[2] Ibid., p. 16.

plain how the same mode of production can have different levels of surplus by identifying what the fundamental element is that increases productivity. In industrial capitalism, Castell argues, the main productive elements are new sources of energy (e.g., steam, electricity, oil) and how effectively they are used throughout production and distribution processes. In informational capitalism the main source of increased productivity comes from the use of "the technology of knowledge generation, information processing, and symbol communication"[3][4]. As the economy shifts its orientation from energy to information as the primary source of surplus value, the creation, accumulation, and use of that information become the organising principles for capitalist activity.

For a large part, the surplus-generating activities of informational capitalism are mediated by software. As a result, the design of that software helps shape how value can be extracted, the quantity of that surplus, and who can appropriate it – for example, by making workers more productive, giving employers more managerial control, or creating new kinds of data that can be commodified. Because software as a mediating artefact plays an important role in the generation and appropriation of surplus value, software bestows power.

1.2 SOFTWARE CONTROL

The way control over software is distributed, and how that distribution is legitimised, depends on the ideology underpinning the mode of development. Julie Cohen argues that the transformation from industrial to informational capitalism was simultaneously accompanied by parallel transformation from liberalism to neoliberalism.[5] A nebulous concept, one popular definition describes it as "political economic practices proposing that human well-being can best be advanced by the maximization of entrepreneurial freedoms within an institutional framework characterized by private property rights, individual liberty, unencumbered markets, and free trade".[6] By successfully advancing the four core policy areas of neoliberalism – deregulation, non-intervention, free markets, and free trade – large, transnational corporations have significantly increased their social and economic power in the past 50 years.[7] Pasquale argues that the simultaenous retreat of the state has created a reality where people are increasingly governed based not on their citizenship and geographic location (*territorial governance*), but by the corporations who mediate their life (*functional governance*).[8] For ex-

[3] Castells, *The Rise of the Network Society*, p. 17.

[4] Castell acknowledges that information plays an important role in other modes of development (and production) as well, but argues that the key difference in informationalism is that surplus is created through the application of information on information itself: knowledge is used to increase the quality and production of knowledge, rather than, say, the production of material goods.

[5] Julie E Cohen (2019). *Between Truth and Power: The Legal Constructions of Informational Capitalism*. Oxford University Press, USA, p.7.

[6] David Harvey (2007). 'Neoliberalism as creative destruction.' In: *The annals of the American academy of political and social science* 610.1, pp. 21–44, p.22.

[7] Terry Hathaway. 'Neoliberalism as Corporate Power.' In: *Competition & Change* ().

[8] Frank Pasquale (2016). 'Two Narratives of Platform Capitalism.' In: *Yale Law & Policy Review* 35, pp. 309–320.

ample, people across the globe that make their living by selling goods on the Amazon Marketplace are governed by a semi-automated system with no separation of power, no rights of appeal or promise of fair trial (inspiring some to fly to the Amazon offices to try and petition its arbitrary decisions[9]). Similarly, regulatory authorities trying to hold Uber accountable for its tax avoidance and illegal licensing were thwarted by Uber's software system Ripley, which allows the company to lock-and-destroy computers remotely.[10] The far-reaching consequences are that the *Rechtsstaat* has been decentered as a provider of human rights, and state-centric institutions of oversight have been replaced by commercial platforms. Control over justice, equality, generality, privacy, publicness – the predicate conditions for a dignified life – is increasingly in the hands of transnational, privatised actors for whom principled governance is an aspiration rather than an obligation [11]. Although technology and software companies are not the only beneficiaries, they are undoubtedly some of the most influential ones.

1.3 SOFTWARE NEGOTIATION

If we agree that software facilitates expressions of power, that control over that software is primarily concentrated in the hands of private actors, *and* that this is undesirable; how do we respond? The normative stance of this dissertation is that *control over software should be distributed and subject to consensus-seeking negotiations.*

Negotiation generally refers to "a process by which two or more parties attempt to resolve their opposing interests",[12] and has been addressed across research disciplines such as psychology, economics, anthropology, sociology, and political science. Given the interdisciplinary nature of the concept, as well as its high level of abstraction, negotiation can be considered *essentially contested*: "concepts the proper use of which inevitably involves endless disputes about their proper uses on the part of their user".[13] What constitute the core characteristics of a negotiation beyond the ambiguous description above depends on who you ask, and providing a strict typology quickly becomes an exercise in disciplinary gate-keeping rather than a way to apply it productively. Ultimately, negotiation is as negotiation does, regardless of whether it meets a particular scholarly definition.

[9] Josh Dzieza (Dec. 19, 2018). 'Prime and Punishment: Dirty dealing in the $175 billion Amazon Marketplace.' In: *The Verge*. URL: https://www.theverge.com/2018/12/19/18140799/amazon-marketplace-scams-seller-court-appeal-reinstatement (visited on 06/21/2020).

[10] Olivia Solon (Jan. 11, 2018). 'Uber developed secret system to lock down staff computers in a police raid.' In: *The Guardian*. URL: https://www.theguardian.com/technology/2018/jan/11/uber-developed-secret-system-to-lock-down-staff-computers-in-a-police-raid (visited on 06/21/2020).

[11] To provide a timecapsule-like example, see the unilateral decision by Google and Apple to only support decentralised architectures for contact tracing apps in response to the COVID-19 pandemic, which has exposed the gaping powerlessness of sovereign states to determine how to use technology in their public health response

[12] Roy J. Lewicki, Bruce Barry, and David M. Saunders (2016). *Essentials of Negotiation*. 6th ed. McGraw Hill. ISBN: 978-0-07-7862466.

[13] W. B. Gallie (1955). 'Essentially contested concepts.' In: *Proceedings of the Aristotelian society*. Vol. 56. Wiley, pp. 167–198.

Extending the above definition of negotiation in general, *software* negotiation in this dissertation is used to refer to *a process by which two or more parties attempt to resolve their opposing interests about the design of a software system*. Although this narrows the concept to a particular context, it is hardly specific enough to shed the essentially contested baggage of negotiation at large. This begs the question: if these theoretical constructs will always be challenged no matter our efforts, why bother trying to develop and clarify them?

The aim of this dissertation is not to try and work towards a single true meaning of the concept of software negotiation, but instead to use it pragmatically and self-reflectively in the pursuit of other goals. My concern with using it as a tool rather than clarifying its essence – with praxis over theory – is also reflected in the title of this document: "negotiating software" signals my focus on the act of effecting a software change, whereas the inverse – "software negotiation" – would place the emphasis on the theoretical.

Other interpretations of software negotiation might appear that focus on a single party rather than two or more; that discuss situations where interests between parties are aligned, rather than conflicting; that target more specifically the initial development or continuous deployment of a system rather than its entire lifecycle. These interpretations will flow from the epistemology, purpose, tradition, and skills of those who use the concept of software negotiation. Human-Computer Interaction (HCI), for example, – the computing research this dissertation mostly speaks to – generally seems to implicitly believe that software problems are a result of well-meaning technologists who misunderstand the complexity of the social world, but that the software can be negotiated by documenting bad designs and suggesting alternatives. Professional computing organisations, such as the Association of Computing Machinery, have mostly responded to the controversies of information-based harms by introducing ethics and codes of conduct. Their underlying approach to negotiation appears to be that individual liberty, moral responsibility, and self-regulation are the best way to address structural issues.

Rather than debate whether these perspectives better reflect the true nature of software negotiation, I welcome any use of the concept that will productively achieve the goals of its authors. Ideally, this openness to pluriformity will help us develop a conceptual toolbox with different approaches to negotiating software that are attuned to different problems, collectively working towards addressing the hegemonic way control over software design is currently distributed.

This dissertation reports on two parallel negotiation projects I have carried out over the course of my doctoral work, which develop two of such conceptual tools: the idea of *negotiable software* and of *negotiation software*.

1.3.1 *Negotiable software*

Negotiable software refers to any software system which inscribes negotiability directly into its design, allowing it to be changed in a substantial and sustainable way by any actor. It represents a principle-based approach and implicitly assumes that negotiability can be translated into concrete technical requirements.

Parallel concepts exist in Human-Computer Interaction and adjacent computing research fields that try to bring some measure of agency to a user over the design of a piece of software. *End-user development* (EUD) is arguably the largest umbrella term for research efforts in this direction. One popular definition describes it as "methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact".[14] The focus on non-professional activity is an important one, and is often combined with the idea that these users have limited technical skills that need to be catered to. Spreadsheets are considered a paradigmatic tool that supports end-user development, and programming-by-example an approach that lowers the threshold for end-users to start developing and adapting software.

Customisability and *configurability*[15] are principles that might support EUD on the lower end of the scale, referring to the parameterisation of a system such that users can choose between various pre-defined alternative designs (most commonly through the settings or preferences options of software). *Tailorability* raises the ceiling a little, and is argued to include modifying the system in use,[16] allowing for substantial changes to the software such as "specializing behavior, or adding functionality".[17] Reflecting the blurry boundaries between all these principles, Mørch[18] actually includes customisability as a sub-concept of tailorability, together with integration (tailoring through "linking together predefined components") and extension ("adding new code"). *Appropriability* is another of those principles that shares considerable overlap with tailorability, although its use by some scholars (e.g., Mackay,[19] Dix,[20] Tchounikine[21]) go beyond the focus on technological features and also consider organisational and institutional factors that support or inhibit how users change software systems.

The popularity of these principles and concepts wax and wane, even if their content or aim does not change all that much. End-user development, although an isolated but cohesive research topic at the flagship CHI conference until 2003,

[14] *End User Development* (2006). Vol. 9. Human-Computer Interaction Series. Springer Netherlands. ISBN: 978-1-4020-4220-1. DOI: 10.1007/1-4020-5386-X. URL: http://link.springer.com/10.1007/1-4020-5386-X.

[15] James R Eagan and John T Stasko (2008). 'The buzz: supporting user tailorability in awareness applications.' In: *Proceedings of the sigchi conference on human factors in computing systems*, pp. 1729–1738.

[16] *End User Development*.

[17] Randall H. Trigg, Thomas P. Moran, and Frank G. Halasz (1987a). 'Adaptability and Tailorability in NoteCards.' In: *Human–Computer Interaction–INTERACT '87*. Elsevier, 723–728. ISBN: 978-0-444-70304-0. DOI: 10.1016/B978-0-444-70304-0.50117-5. URL: http://linkinghub.elsevier.com/retrieve/pii/B9780444703040501175.

[18] Anders Mørch (1997). 'Three levels of end-user tailoring: Customization, integration, and extension.' In: *Computers and design in context*. Ed. by Morten Kyng and Lars Mathiassen. MIT Press, pp. 51–76.

[19] Wendy Mackay (1990). 'Users and customizable software: A co-adaptive phenomenon.' PhD thesis. Massachusetts Institute of Technology.

[20] Alan Dix (2007). 'Designing for appropriation.' In: *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 2*. British Computer Society, 27–30. URL: http://dl.acm.org/citation.cfm?id=1531415.

[21] Pierre Tchounikine (2017). 'Designing for Appropriation: A Theoretical Account.' In: *Human–Computer Interaction* 32.4, 155–195. ISSN: 0737-0024, 1532-7051. DOI: 10.1080/07370024.2016.1203263.

has largely disappeared from that stage since then. Newly emerging terms that try to address some of the same concerns, if with perhaps a different inflection, include *no-code*, *low-code* and *live programming*. No-code and low-code describe development environments for interactive systems that do not require manual coding but instead support more graphical or automated code generation; again to facilitate the creation or adaptation of software by a wider group of people. Live programming refers to the act of changing software on-the-fly during execution time.[22] Environments supporting such liveness help shorten the feedback loop between authoring and using software systems, which also opens up the possibility that anyone could alter a running system instead of just those with access to the source code and the necessary compilers.

These concepts represent a tenacious and messy patchwork of research efforts whose distinctions are not entirely clear, but which all try to work towards allowing end-users to adjust the design of their software systems. The principle of negotiability overlaps with these other concepts to some extent but adds the explicit concern for power dynamics and conflict, which are not absent from existing work but also not necessarily systematically included. Negotiability connotes a process of discussion and compromise between actors, which inherently includes some reflection on how those different actors relate to each other and how that might affect the process of negotiation. These dynamics can be the result of things such as who actually has access to code, who has the technical skills to make adjustments, who is perceived to have the ownership of a system, and how the different actors are placed in existing social hierarchies. The goal of the negotiability principle as a conceptual tool is to extend previous research on how users adapt software to their local needs, and explicitly recognise that the overall power dynamics around technology have changed dramatically since the introduction of personal computing devices in the 1980s.

1.3.2 *Negotiation software*

Negotiation software refers to any system which supports the negotiation activities of two or more parties. It represents a process-based approach, and is analogous to using technology to facilitate any other domain-specific tasks such as creative design,[23] air-traffic control,[24] or patient record management.[25] In the context of this thesis, negotiation software is oriented towards supporting processes that try to change the design of software, but negotiation software could of course also be used in contexts such as international conflict resolution, collective bargaining, or hostage situations.

[22] Steven L Tanimoto (2013). 'A perspective on the evolution of live programming.' In: *2013 1st International Workshop on Live Programming (LIVE)*. IEEE, pp. 31–34.

[23] Jonas Frich et al. (2019). 'Mapping the landscape of creativity support tools in HCI.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–18.

[24] R. R. Harper, J. A. Hughes, and D. Z. Shapiro (1990). 'Harmonious Working and CSCW: Computer Technology and Air Traffic Control.' In: *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. NLD: North-Holland Publishing Co., 225–234. ISBN: 044488811X.

[25] Trisha Greenhalgh et al. (2009). 'Tensions and paradoxes in electronic patient record research: A systematic literature review using the meta-narrative method.' In: *The Milbank Quarterly* 87.4, pp. 729–788.

There are two strands of computing-related research relevant for the concept of negotiation software: the use of software to support processes of change of any kind, and the use of any kind of method to change the design of software. The first strand – how technology can mediate processes of change – is explored primarily in the context of participatory democracy activities such as collective actions, grassroots movement building, and community organising. Social media technology specifically has been attributed a significant role in democratic activities, such as how protests start and develop. These studies generally argue that social technologies make protest much easier and faster to organise, but that this method does not help build the infrastructure necessary to sustain a movement in the long-term.[26] Other studies more narrowly reflect on the use of software for specific types of movements. Dencik and Wilkin, for example, discuss digital activism in international labour movements, and suggest that incorporating technology-based strategies might help address the core challenges trade unions currently face, such as decline in membership, decreasing labour power, and a decentering of the workplace as a place to organise.

An inspirational article by Abebe et al.[27] published at the recently-founded Fairness, Accountability, and Transparency conference proposed four high-level ways that computing could contribute to social change: as a *diagnostic*, by measuring social problems using computational methods; as a *formaliser*, by explicitly defining social problems when translating them into computational models; as a *rebuttal*, by showing the limits of technological implementations, and; as *synecdoche*, by using the public's fascination with computers as a vehicle to bring long-standing issues back into the mainstream. These types of studies give us an insight into how software can be used in negotiation processes at various levels of abstraction for various democratic projects.

The second strand of research discusses how to bring change to software, through various methods and channels. A variety of different traditions might fall under this category, depending on how widely we consider 'change' and 'methods'. Thinking broadly, participatory design (PD) is one of these traditions, founded in the 1970s to help workers and unions have some influence on computing solutions introduced by their employers. While still a lively area of research with its own dedicated conference, some of the founding scholars have criticised contemporary approaches for being too focused on small, relatively unimportant issues,[28] and only involving the final users of the technology to elicit requirements and test the usability, rather than in two-way, long-term processes.[29] The PD approach represents a broad view on bringing change to software, as it includes many other concerns as well.

[26] Zeynep Tufekci (2017). *Twitter and tear gas: The power and fragility of networked protest*. Yale University Press.

[27] Rediet Abebe et al. (2020). 'Roles for computing in social change.' In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 252–260.

[28] Susanne Bødker and Morten Kyng (2018). 'Participatory design that matters—Facing the big issues.' In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 25.1, pp. 1–31.

[29] Claus Bossen, Christian Dindler, and Ole Sejer Iversen (2012). 'Impediments to user gains: experiences from a critical participatory design project.' In: *Proceedings of the 12th Participatory Design Conference: Research Papers-Volume 1*, pp. 31–40.

The efforts of technologists to communicate with policymakers represents a more targeted approach to bringing change to software. The HCI community has had little involvement with public policy communities in the past,[30] but this has been changing in recent years, with researcher writing white papers, trying to get more policy experience through work placements, and developing graduate courses to teach the necessary skills to future generations.[31] These new moves are not without challenges, as what is considered publishable research is not always valued as acceptable evidence by policymakers, and developing the right communication strategy around that evidence takes time to develop.[32]

Negotiation software in this dissertation is the combination of these two strands: the use of computational tools to support negotiation processes directed at changing a software's design. The goal of this conceptual tool is to explore how software can improve the effectiveness of existing negotiation strategies, what new strategies it might open up, and in what situations it might do more harm than good.

To answer the question posed at the start of this chapter, we should care about negotiating software because, since the 1970s, digital technologies increasingly determine "who gets what, when, and how",[33] and control over those technologies has concentrated in the hands of private corporations who have no mandate to ensure they benefit the people whose life it mediates. We should consider the demand that control over software is distributed and subject to consensus-seeking negotiations as a way to address this current power asymmetry. Two ways in which this could be operationalised, and which this dissertation will explore, is by embedding the principle of negotiability directly in software, and by leveraging software to support existing processes of negotiation.

[30] Jonathan Lazar et al. (May 2016). 'Human–Computer Interaction and International Public Policymaking: A Framework for Understanding and Taking Future Actions.' In: *Found. Trends Hum.-Comput. Interact.* 9.2, 69–149. ISSN: 1551-3955. DOI: 10.1561/1100000062. URL: https://doi.org/10.1561/1100000062.

[31] Anne Spaa et al. (2019). 'Understanding the Boundaries between Policymaking and HCI.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–15.

[32] Ibid.

[33] Harold D. Lasswell (1936). *Politics: Who Gets What, When, How*. Whittlesey House.

2.

RESEARCH QUESTIONS & DISSERTATION STRUCTURE

The overarching research question of this dissertation is:

How can the design of software be negotiated?

The dissertation is separated into two parts. Each part focuses on a different kind of software to negotiate, and uses a different theory of change to inform the process.

Part I targets *the application*: the paradigmatic model of commodified, turn-key software for end-users. Its theory of change is *design-centric*, which assumes that improvements to software originate from a better understanding of the context in which it is used, followed by a close collaboration with the intended users to develop and implement an alternative design. Its goal is to produce *negotiable software*, technology which embeds the principle of negotiation directly into its design.

Part II targets *consent management platforms*: pop-up interfaces that purport to capture the consent of website visitors, which have appeared in response to EU data protection regulation. Its theory of change is *regulation-centric*, which assumes that changes to software are the result of the implementation and enforcement of state-based regulation. Its goal is to develop *negotiation software*, technology which supports existing processes of negotiation.

Part I: negotiable software

Chapter 5 provides a historic overview of application software, starting from the mainframes of 1940s and ending with the proliferation and stabilisation of the microcomputer industry in the 1990s. It traces the origins of the term application software and how its design evolved. The primary goal of this chapter is to demonstrate that the application model is a construct. The main research question is: **How was the current dominant design of application software constructed?**

Chapter 7 gives a broad outline of contemporary application use by Danish knowledge workers. Based on a representative survey, it details which applications they use, their digital competences, and their overall customisation strategies. It shows how the Danish knowledge industry is dominated by a handful of US software and that they use mostly turn-key rather than personalised software.

The research question is: **What is the contemporary landscape of application use by Danish knowledge workers?**

Chapter 8 counter-balances the quantitative overview of the previous chapter and describes the lived experience of Danish knowledge workers with application software. It shows how i) their economic value is intertwined with data and skills related to specific applications; ii) their access to this value is systematically jeopardised in collaboration due to the different application practices, preferences, and proficiencies of other stakeholders; and iii) they mitigate the costs of this compromise through cross-application collaboration strategies. The main research question is: **How does the application model impact labour conditions of Danish knowledge workers?**

Chapter 9 switches from empirical descriptions to normative interventions, and presents a negotiable software system: Codestrates. Rather than separating multimedia content from code, Codestrates allows both to be written in the same perceptual space, blurring the distinction between development and use of interactive systems. Its aim is to demonstrate the technical feasibility of an alternative software model where its design can be altered on the fly, collaboratively, from within itself. The main research question is: **What does an alternative software model look like that is technologically negotiable?**

Chapter 10 brings the software model back into a real-world context, and describes a participatory design process with a laboratory of biomolecular nanoscientists. It builds on the Codestrates platform and implements a negotiable software system to support the computational design of RNA structures. It uses this experience to reflect on the principles of negotiability. The research question is: **How can negotiable software better support the work practices of a particular group of biomolecular nanoscientists?**

Part II: negotiation software

Chapter 13 provides a historic overview of consent management interfaces, starting from the invention of internet tracking technologies in the mid-1990s to the implementation of the General Data Protection Regulation in 2018 and the subsequent boom of third-party consent services. The research question is: **Why did pop-up interfaces become the model for consent management and the exercise of digital rights on the web?**.

Chapter 14 gives a large-scale overview of the designs of consent management platforms, collected by scraping the most popular third-party services used on the top 10,000 most popular websites in the UK. It shows how the vast majority of these interfaces do not comply with the design specifications required by EU law for legally-valid consent. It also reports on a controlled experiment to demonstrate how the design of consent interfaces affect the answers submitted by internet users. It directly speaks to the requirement for consent to be "freely given" as described in article 4(11) of the GDPR. It shows how the notification style does not affect people's consent choices, but that the ordering of buttons and option granularity does. The research question is: **What is the contempo-**

rary landscape of consent management interfaces on the web and how does it impact people's consent choices?

Chapter 15 reports on my efforts to use the data about consent pop-ups gathered using a web-scraper to reach out to news media and regulatory authorities. It describes how the Danish Data Protection Authority made a verdict in an ongoing consent interface complaint and released new guidance for companies that directly addressed the non-compliant designs highlighted in Chapter 14. The research question is: **How can automated compliance monitoring software be used to support existing regulatory processes of negotiation?**

Chapter 16 describes an individualistic approach to negotiating consent management platforms, as a safe-guard for when regulatory action falls short. It introduces a browser extension which automatically answers consent pop-ups, subverting their manipulative designs and allowing people to take direct action. The research question is: **How can adversarial interopability be used to negotiate software designs?**

2.1 A PERSONAL REFLECTION ON THE DISSERTATION CONTENT AND STRUCTURE

A doctoral dissertation is the final product of a multi-year intellectual project, but its design and structure are shaped by more things than just scholarly concerns about what the best way is to answer a particular research question. In this section I will briefly reflect on how other constraints – personal, institutional, and national – have affected my work.

In Denmark, a PhD is usually a three-year, fully-funded, full-time salaried position (37 hours per week). In line with the Ministerial Order on the PhD Degree Programme, at Aarhus University a PhD position includes 1) independent research work under supervision; 2) 840 hours of departmental work (i.e., teaching, conference organising); 3) PhD courses or similar activities corresponding to 30 ECTS credits (i.e., 840 hours); 4) a stay at a foreign research environment of 2-6 months; and, 5) submitting a PhD dissertation.[1] At a total of 4.929 working hours over the course of three years, the 1680 hours for departmental work and accredited courses take up one third. Accounting six months for the stay at a foreign research environment, and another six months for writing the dissertation, this leaves roughly one year for other independent research work.

There are two ways to be hired as a PhD student at Aarhus University: either by applying through the university's open call with your own project, or by applying for a specific call to a pre-defined project. My PhD falls under the second category, and was advertised under the title "The Present and Future of Digital Tools and Materials in Design and Knowledge Work", funded by Aarhus Universitets Forskningsfund. The original funding application stated that "[t]he PhD student [...] will primarily focus on studying and conceptualising present and future prac-

[1] Ministry of Science, Innovation and Higher Education (2013). *Ministerial Order on the PhD Degree Programme at the Universities and Certain Higher Artistic Educational Institutions*. https://www.retsinformation.dk/eli/lta/2013/1039.

tices of computer use in design and knowledge work with cases in academia (e.g. at Nanoscience or Interacting Minds) and private companies (e.g. DesignIt)".

Aarhus University accepts two kinds of dissertation formats: the monograph – a "dissertation written independently by the PhD student" – and a publication-based thesis – "a collection of several academic that are related in content and/or methodology and where the results obtained in course of the PhD programme are presented and possibly published, either by the PhD student alone or by the student together with other authors".[2]

These national and institutional conditions affected the content, direction, and final structure of my dissertation in a number of ways.

I chose to do a dissertation based on publications, since the research group I was employed at was situated in the field of Human-Computer Interaction, where papers are the common currency rather than monographic books more common in other fields. Having calculated the number of hours allotted for independent research at the beginning of my PhD, I decided to aim for one paper submission per year, which naturally had an impact on the length and scope of the projects I imagined. I decided that these papers would have to make sense on their own if I wanted to be able to publish them, rather than only in the context of a larger dissertation. They would also require a certain level of independence, since previous experience had told me research projects rarely work out exactly as planned, and relying on finding something in one project to justify a follow-up study would be much more fragile and encourage premature optimisation. My supervisor, associate professor Clemens Nylandsted Klokmose, also frequently mentioned he considered a PhD "like getting a driver's license" for researchers. I interpreted this to mean that I should treat the PhD more as an educational opportunity than already a commitment to a specific professional identity, and use it primarily to learn how research worked, what the rules of academia were, and explore what kind of scholar I would like to be.

All these things combined meant that my general attitude to the PhD was that I would try to work towards a coherent dissertation and answer a main research question, but generally prioritise self-contained papers that would teach me a new skill, method, theory, or concept. That way, even if the paper was an "academic" failure, it would still be a success seen through the perspective of the PhD as a "driver's license". As a result, the chapters in this dissertation include a diverse set of methods, epistemologies, topics, and participants, chosen not just based on scholarly considerations but also my interests, pragmatic constraints, and available opportunities.

In the first year of my PhD, I mostly worked on research projects already ongoing in the group and on topics and questions included in the project description of the "specific call" I was hired under. For example, the research reported on in Chapter 9 was already well-underway when I arrived at Aarhus University. The

[2] Aarhus University Graduate School of Arts (2012). *Rules for the PhD Programme at the Graduate School, Art*. https://phd.arts.au.dk/fileadmin/phd.arts.au.dk/AR/Generelle_retningslinjer_UK_1-11-2012.pdf.

technology had been built, an evaluation was being conducted, and my role was to contribute to framing the contribution, writing the paper, recording the video, rebut the reviews, and prepare the final submission. The research reported on in Chapter 8 was inspired by my supervisor's suggestion that I should start by "getting my hands dirty" (advice I was happy to follow). It was a replica of my Master's thesis methodologically, but instead applied to the research questions in the PhD funding proposal. Since I had just moved to Denmark and did not speak the language or have a large network, the recruitment strategy for participants was shaped as much by whoever I could convince to lend me their time as by a considered idea of which knowledge workers I was most interested in. The project in Chapter 10 was a combination of the outcomes of these two previous projects – the Codestrates platform and the focus on the harms of application-centric computing – in collaboration with a nanoscience research group that had already been included as a partner in the funding application. Naturally, this group of knowledge workers was much more specific than the earlier study, but the opportunity to do long-term participant observation won out over the worry about demographic consistency between projects.

The early months of the second year of my PhD was spent writing up a first version of what would eventually become the paper in Chapter 10. I struggled trying to connect my observations about how these knowledge workers really used technology with the framing of the funding proposal and the initial idea of building computational interventions. Most technology-related problems these workers had were mundane, structural issues such as insufficient funds to buy software, or the slow processing speeds of their computers, or the general lack of commercial software for their niche area of research. When the paper was rejected for the 2019 cycle of the ACM CHI Conference on Human Factors in Computing Systems, I had become dissatisfied with my (and HCI's in general) approach to computing-related problems, which I started to see more as technological solutionism than actually helpful contributions.

One of my responses to this dissatisfaction was to try to do more generalisable research in the form of a large-scale survey, the results of which are reported in Chapter 7. This was also the time that I was expected to arrange a research stay at a foreign institution. One of my other responses was to use this opportunity to explore the policy-side of technology design by joining Tim Berners-Lee's Decentralised Information Group at Massachusetts Institute of Technology, supervised by Lalana Kagal, who had previously been involved in policy-aware technology projects. To strengthen the link to my earlier HCI work, I also collaborated with David Karger's Haystack group, which published at HCI venues and was situated on the same floor at the Computer Science and Artificial Intelligence Lab. Trying to find a research project at the intersection of policy, HCI, MIT, Aarhus University, my interests, and the interests of the two professor I was visiting, resulted in the publication in Chapter 14. A not-insignificant factor in the decision was the purely opportunistic consideration that the General Data Protection Regulation had just gone into effect, and that being a European citizen meant I was seen as qualified to work on this subject.

The results of that study exposed the limitations of technology policy as well (without effective enforcement, regulation remains toothless), so in my third year of the PhD I decided to find a middle-ground: technological interventions to help with policy-related problems, the results of which are described in Chapter 16 and Chapter 15. The remaining time of the PhD contract, as well as an additional six months during which I taught at Aarhus University to supplement my income, I focused on connecting these various projects, and wrote the chapters that would form the necessary bridge.

These practical constraints shaped my doctoral work in unanticipated ways and for reasons beyond scholarly concerns. It also helped me develop a nuanced perspective on the intersection between HCI and technology policy; on the strengths and weaknesses of regulatory interventions and silver software bullets; on the epistemologies of interviews, observations, surveys, controlled experiments, historiographies, and computational methods; and, perhaps most constructively, on how to balance personal motivations and academic contributions.

3.

PUBLICATIONS

In addition to new material, this dissertation also includes content from the following list of previously published documents.

Roman Rädle, **Nouwens, Midas**, Kristian Antonsen, James R Eagan, and Clemens N Klokmose (2017). 'Codestrates: Literate computing with webstrates.' In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. Quebec City, QC, Canada: Association for Computing Machinery, pp. 715–725. DOI: 10.1145/3126594.3126642. URL: https://doi.org/10.1145/3126594.3126642

Nouwens, Midas and Clemens Nylandsted Klokmose (2018). 'The Application and Its Consequences for Non-Standard Knowledge Work.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173973. URL: https://doi.org/10.1145/3173574.3173973

Nouwens, Midas, Marcel Borowski, Bjarke Fog, and Clemens Nylandsted Klokmose (2020a). 'Between Scripts and Applications: Computational Media for the Frontier of Nanoscience.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376287. URL: https://doi.org/10.1145/3313831.3376287

Nouwens, Midas, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal (2020b). 'Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376321. URL: https://doi.org/10.1145/3313831.3376321

Part I

Negotiable Software

4.

INTRODUCTION

Artefacts have politics,[1] code is law,[2] and technological mediation embeds intentionality:[3] three slogans used across different disciplines which all get to the same point. The design of the non-human objects embed properties which help regulate how we behave and perceive the world. While this framing is more commonly used to analyse and critique existing artefacts produced by others, it also implies that we can imbue our own politics, law, and intentionality into the artefacts that we design.

Part I of this dissertation describes the process of developing *negotiable software*, i.e., software that is designed to explicitly and inherently support being changed through multi-stakeholder engagement.

4.1 SOFTWARE: APPLICATIONS

Software can take many forms, but arguably the most well-known shape of software is the application: an isolated piece of software installed on a computer that operates on certain data types using a set of predetermined programs through a graphical interface. To explore negotiable software at large, this dissertation focuses on application software in particular.

Application-centric computing has dominated human-computer interactions for the past forty years, to the point that it has achieved Weiserian-levels of invisibility. The vast majority of users experience computation solely through applications: they use computers not as programmable universal devices, but as "a specific machine with a specific behavior for a specific purpose"[4] (e.g., a spreadsheet, a word processor, an email client, a web browser). This model of software is a construct with its own set of embedded politics, shaped by the process of commodification and mass-marketisation it went through between the 1950s

[1] Langdon Winner (1980). 'Do artifacts have politics?' In: *Daedalus*, pp. 121–136.

[2] Lawrence Lessig (1999). *Code and other laws of cyberspace*. Basic Books. ISBN: 978-0-465-03912-8.

[3] Peter-Paul Verbeek (2005). *What things do: Philosophical reflections on technology, agency, and design*. Penn State Press.

[4] Bengt Goransson et al. (1987). 'The Interface is Often Not the Problem.' In: *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*. CHI '87. Toronto, Ontario, Canada: ACM, pp. 133–136. ISBN: 0-89791-213-6. DOI: 10.1145/29933.30872. URL: http://doi.acm.org/10.1145/29933.30872.

and 1990s in the United States. As a result, it is negotiable in a specific, limited way that mostly favours the developers and their business models, rather than users/consumers/citizens and their rights. In general, the application is globally distributed but its design is locally controlled; its code is often proprietary rather than freely available; it facilitates customisation through sand-boxed plugins and scripts rather than run-time reprogramming; it updates autocratically rather than democratically; it does not interoperate with other software. Exceptions exist, but the dominant design of application software favours unilateralism over negotiability.

4.2 CONTEXT: KNOWLEDGE WORK

Applications are used in many contexts, but one of the earliest industries and successful markets for software applications was *knowledge work*: jobs that involve the creation and distribution of (digital) information through non-routine, creative, and abstract thinking. To explore negotiable applications, Part I of this dissertation focuses on contemporary knowledge work and workers.

Although the concept of knowledge work suffers from policy evangelism and lacks an operationalised definition[5], at the most abstract level it refers to any labour that uses existing information in flexible and innovative ways to produce new information from which value can be extracted. The knowledge economy itself emerged in the 1990s-2000s, when capital returns on mass-produced physical goods slowed down and global competition increased. Many countries committed to the idea of "knowledge" as the new, more efficient asset that would guarantee continued economic growth. Examples of knowledge-based capital include things such as patents, intellectual property, brand-equity, innovation research, and, of course, software.

In knowledge economies, the knowledge *worker* – offensively called "human capital" – has become the most in-demand commodity, as most of the additional surplus value is created when the worker has more knowledge and uses it more effectively. Application software has played a facilitative role in this increased efficiency from the beginning. The application as a model of software first emerged during the late 1970s and early 1980s in the United States, and only became a commercially successful product because it managed to capture the imagination of large corporations and white-collar office workers. A considerable reason why the knowledge economy became a viable alternative to the manufacturing economy was because computers increased the rate at which information could be produced and processed by orders of magnitude, and because application soft-

[5] EU and OECD white papers have variously attempted to capture knowledge work by describing it based on the sector or industry they work in, the activities common in their work, the level of education required, or their occupation category, but none have allowed governments and businesses to measure and intervene effectively in this type of labour. See (Ian Brinkley, Michelle Mahdon Rebecca Fauth, and Sotiria Theodoropoulou [2009]. *Knowledge workers and knowledge work: A knowledge economy programme report*. Work Foundation) for a discussion

ware made it possible for humans to leverage that capability at scale. To this day, the more knowledge intensive industries continue to be the most digitised.[6]

To explore the idea of negotiable application software, we will be going back to the original context for which it was developed, and which could not exist without it.

[6] Jacques Bughin et al. (2016). 'Digital Europe: Pushing the frontier, capturing the benefits.' In: *McKinsey Global Institute*.

5.

A BRIEF HISTORY OF THE APPLICATION; OR, HOW THE COMMODIFICATION OF SOFTWARE SHAPED ITS NEGOTIABILITY

5.1 INTRODUCTION

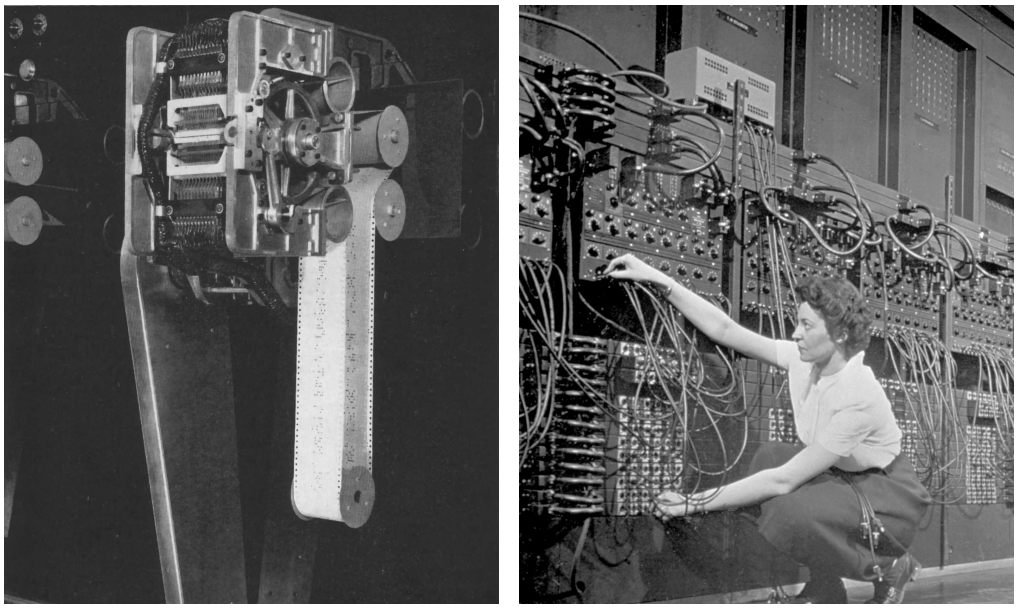
The "application" is not a natural, inevitable design of software, but rather a particular construct profoundly shaped by the process of commodification it went through in the United States between the 1950s and the 1990s. Despite its ubiquity, the application model of software is almost wholly ignored as a topic of inquiry by HCI research and the history of computing communities. As a result, it is difficult to imagine our everyday interaction with computers mediated by anything other than applications, or see a path towards a different model in the future.

The aim of this chapter is to trace how the *negotiability* of software has changed over time by describing the history of the application model from the 1950s until the 1990s. It will follow the social and economic construction of this technology by looking at how, at different points in time, software was defined, what its purpose was imagined to be, who was included in its development, and what activities it supported or inhibited.

The words used to distinguish between different designs and commodifications of software, even the word software itself, are anachronisms that did not exist or refer to the things we might use it for today. For the sake of clarity, however, I will use software ahistorically as a catch-all term for non-material computer instructions, in conjunction with the actual words used to describe the artefact at the time whenever possible.

5.2 THE MATERIAL CREATION OF SOFTWARE

Before the 20th century, there was no need to distinguish between 'hardware' and 'software': a machine's execution instructions were largely hardcoded into the device's design. A first *conceptual* distinction between a device and its instructions was made in the late 1930s and early 1940s with the invention of the universal, general purpose, programmable computer: a computer could now (in



(a) Tape reader of the ASCC. **(b)** Ruth Lichterman wiring the ENIAC.

Figure 1. Machine instructions for mechanical and electronic mainframe computers.

principle) be adapted to calculate anything, rather than just those specified in its original design. For mechanical and electromechanical mainframe computers (e.g., the Automatic Sequence Controller Calculator at Harvard University), these instructions were fed to the machine through spools of punched hole paper tapes (fig. 1a). In fully electronic mainframe computers (e.g., the Electronic Numerical Integrator And Computer), they were provided by connecting plugs to sockets, flipping switches, and turning dials (fig. 1b). However, in both cases the instructions were still firmly rooted in the physical, as using the machine for another "general purpose" still required rewiring or new paper tapes.

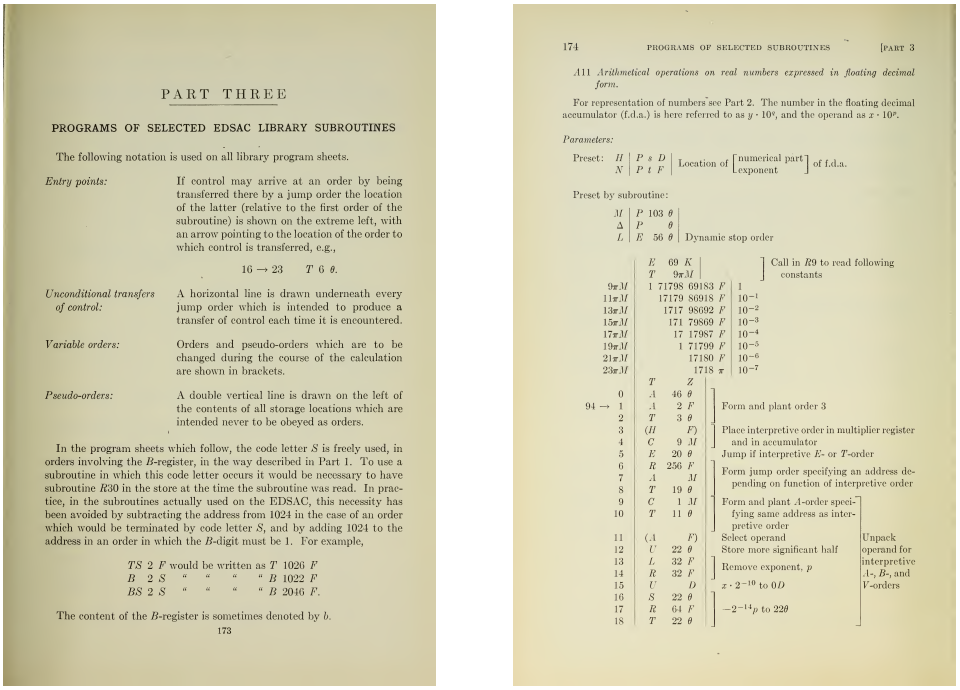
The first *material* distinction between the computer and its instructions was created in 1948-1949 through the successful implementation of stored-program architectures,[1] which allowed calculations to be executed from an electronic memory. For the first time, the device's instructions were purely electronic signals, and the physical state of the computer no longer represented the program it was calculating. To get the instructions into the electronic memory still required feeding them to the machine through punched paper cards or magnetic tapes, however, so this material separation was at least partly theoretical rather than experiential.

Software for these mainframe computers was written by the researchers working on the technology. In the first textbook on programming published in 1951 – "The Preparation of Programs for an Electronic Digital Computer" by Wilkes, Wheeler, and Gill; often referred to as just WWG[2] – more than half of the first edition of the book (85 out of 164 pages) contained the software developed by

[1] Nick Metropolis and Jack Worlton (1980). 'A Trilogy on Errors in the History of Computing.' In: *Annals of the History of Computing* 2.1, pp. 49–59.

[2] Martin Campbell-Kelly (2011). 'In praise of 'Wilkes, Wheeler, and Gill'.' In: *Communications of the ACM* 54.9, pp. 25–27.

the authors and their team working with the Electronic Delay Storage Automatic Calculator (EDSAC) (fig. 2).



PART THREE

PROGRAMS OF SELECTED EDSAC LIBRARY SUBROUTINES

The following notation is used on all library program sheets.

Entry points: If control may arrive at an order by being transferred there by a jump order the location of the latter (relative to the first order of the subroutine) is shown on the extreme left, with an arrow pointing to the location of the order to which control is transferred, e.g.,

16 → 23 T 6 θ.

Unconditional transfers of control: A horizontal line is drawn underneath every jump order which is intended to produce a transfer of control each time it is encountered.

Variable orders: Orders and pseudo-orders which are to be changed during the course of the calculation are shown in brackets.

Pseudo-orders: A double vertical line is drawn on the left of the contents of all storage locations which are intended never to be obeyed as orders.

In the program sheets which follow, the code letter *S* is freely used, in orders involving the *B*-register, in the way described in Part 1. To use a subroutine in which this code letter occurs it would be necessary to have subroutine *R*30 in the store at the time the subroutine was read. In practice, in the subroutines actually used on the EDSAC, this necessity has been avoided by subtracting the address from 1024 in the case of an order which would be terminated by code letter *S*, and by adding 1024 to the address in an order in which the *B*-digit must be 1. For example,

TS 2 *F* would be written as *T* 1026 *F*
B 2 *S* " " " " *B* 1022 *F*
BS 2 *S* " " " " *B* 2046 *F*.

The content of the *B*-register is sometimes denoted by *b*.

173

174 PROGRAMS OF SELECTED SUBROUTINES [PART 3

*A*11 *Arithmetical operations on real numbers expressed in floating decimal form.*

For representation of numbers see Part 2. The number in the floating decimal accumulator (f.d.a.) is here referred to as $y \cdot 10^q$, and the operand as $x \cdot 10^p$.

Parameters:

Preset: *H* | *P s D* | Location of [numerical part / exponent] of f.d.a.
N | *P 1 F* |

Preset by subroutine:

M | *P* 103 θ |
Δ | *P* θ |
L | *E* 56 θ | Dynamic stop order

	E 69 *K*		Call in *R*9 to read following constants
	T 9π*M*		
9π*M*	1 71798 69183 *F*	1	
11π*M*	17179 86918 *F*	10^{-1}	
13π*M*	1717 98092 *F*	10^{-2}	
15π*M*	171 79869 *F*	10^{-3}	
17π*M*	17 17987 *F*	10^{-4}	
19π*M*	1 71799 *F*	10^{-5}	
21π*M*	17180 *F*	10^{-6}	
23π*M*	1718 π	10^{-7}	
	T *Z*		
0	Δ 46 θ		
94 → 1	Δ 2 *F*		Form and plant order 3
2	*T* 3 θ		
3	(*H* *F*)		Place interpretive order in multiplier register and in accumulator
4	*C* 9 *M*		
5	*E* 20 θ		Jump if interpretive *E*- or *T*-order
6	*R* 256 *F*		Form jump order specifying an address depending on function of interpretive order
7	Δ *M*		
8	*T* 19 θ		
9	*C* 1 *M*		Form and plant *A*-order specifying same address as interpretive order
10	*T* 11 θ		
11	(Δ *F*)		Select operand — Unpack operand for interpretive Δ-, *B*-, and *V*-orders
12	*U* 22 θ		Store more significant half
13	*L* 32 *F*		Remove exponent, p
14	*R* 32 *F*		
15	*U* *D*		$x \cdot 2^{-10}$ to 0*D*
16	*S* 22 θ		
17	*R* 64 *F*		$-2^{-14}p$ to 22θ
18	*T* 22 θ		

Figure 2. The first two pages of the program library included in Wilkes, Wheeler, and Gill's *The Preparation of Programs for an Electronic Digital Computer*.

They did not use the word software to describe their code, but instead referred to each self-contained calculation as a subroutine, and a collection of subroutines as a program. Considering the huge effort required to write correct code, the authors advised writing subroutines such that they could be reused, and described how they organised them into a code library. Interestingly, the publication of the book boosted the popularity of the EDSAC hardware design. Because of the lack of computer standardisation, software and hardware were still tightly coupled. To take advantage of the considerable labour represented by the WWG software library, other computers matched the EDSAC (e.g., Japan's first electronic computer, the Tokyo Automatic Computer[3]).

5.3 THE COOPERATIVE DESIGN OF SOFTWARE

While the mainframe computers of the 1940s were research prototypes or military equipment, the 1950s saw the rise of the computer as a commercial product. For example, IBM, with decades of success selling tabulating machinery to clients such as the the US Government and the Third Reich[4], announced its first foray into the commercial computing market in April 1952: the IBM 701 Electronic

[3] Campbell-Kelly, 'In praise of 'Wilkes, Wheeler, and Gill',' p. 2.

[4] From the start, computers have been inextricably linked to warfare, population control, and crimes against humanity.

Figure 3. The IBM 701 Electronic Data Processing Machine.

Data Processing Machine (fig. 3). While the hardware industry developed, the program code required to use the computer remained uncommodified, and still largely the prerogative of the computer user to develop. When the first eighteen customers received the IBM 701[5], it did not come with any software except for a primitive assembler and some "utilities" such as a one-card, memory clearing program.[6]

Almost from the beginning, the skills and time required to develop software have been the main constraints determining the design of the technology and the direction of its industry. The labour cost of programming the early mainframes was so high (often upwards of a year's rental of the device) that IBM considered it a major threat to the future commercial success of computers: organisations were reluctant to buy a new computer and have to redevelop their entire library of programs. With the launch of the IBM 704 on the horizon, an IBM sales manager in the Los Angeles area decided to organise a meeting between all customers of the 701 to discuss whether a collaborative software development effort would be possible to reduce the individual burden. This approach was a profound shift from the normal, internal way software was developed at the time, and no small undertaking, as it included companies that were each other's direct competitors (e.g., RAND Corporation, Lockheed Aircraft Corporation, North American Aviation Inc.), with a history of poaching each other's programmers. After several rounds of drinks, the November 1952 meeting would later be known as the suc-

[5] The first client to receive the 701 was the Los Alamos Scientific Lab in New Mexico, working on the development of nuclear weapons.

[6] Martin Campbell-Kelly (2004). *From airline reservations to Sonic the Hedgehog: a history of the software industry*. MIT press, p. 30.

cessful inauguration of the "Digital Computer Association" (DCA), colloquially known as the Drunkard's Computer Association.[7]

The DCA's efforts resulted in the first cooperative design of a piece of software for the 701 called, appropriately, PACT: the Project for the Advancement of Coding Techniques. Although the compiler was an impressive accomplishment and a better piece of software than anything offered by IBM, the DCA was convinced that "[t]he spirit of cooperation between member organizations and their representatives during the formulating of PACT-I has been one of the most valuable resources to come from the project. It is essential that this spirit of cooperation continue with future project plans".[8]

And continue it did. When IBM announced the 704 in May 1954, the groups involved in PACT started discussing a new joint effort to standardise assemblers and avoid the redundancy of each programming the same set of mathematical subroutines (e.g., numerical inversions) and utility programs (e.g., input-output programs). This eventually culminated in the founding of SHARE in 1955, the very first "User's organization" of the computing industry. Its explicit goals were "1) the standardization of machine language and certain machine practices, 2) the elimination of redundant effort expended in connection with use of the computer, 3) the promotion of inter-installation communication and 4) the development of a meaningful stream of information between the user and the manufacturer".[9] Within two years, the SHARE network of IBM 704 installations grew from eighteen to forty-seven. A Policy Committee was appointed to direct the efforts and distribute programming assignments to members. Once finished, these were made available to all other members as part of the SHARE distributions, disseminated via post by the SHARE Program Library Agency as magnetic tapes or punched cards (if not exceeding more than 1000 cards).[10]

The activities of the DCA and the organisations that emerged out of it offer a window to a fundamentally different mode of software production. The choice of the names PACT and SHARE both strongly signal commitments to plurality, cooperation, and compromise. A point of pride among the founders of SHARE was the strong spirit of collaboration between the representatives of highly competitive companies.[11] In each case, the defining principles of the networks were not just technical, but codified norms about what it meant to be part of the cooperative. As stated in the "Obligations of a SHARE Member" included in the SHARE Reference Manual for the IBM 704, "[t]he principle obligation of a member *is to have a cooperative spirit*"(emphasis original).[12] The obligations also included the

[7] Mary Brandel (1999). '1955: IBM customers form the first computer user group.' In: *CNN*. URL: http://edition.cnn.com/TECH/computing/9905/05/1955.idg/.

[8] Paul Armer (1956). 'SHARE - A Eulogy to Cooperative Effort.' In: *Annals of the History of Computing 2*.

[9] Steve Guendert (2011). 'Mainframe History and the First User's Groups (SHARE).' In: p. 2. URL: https://www.cmg.org/wp-content/uploads/2011/05/m_79_5.pdf.

[10] SHARE Program Library Agency (1977). 'User's Guide and Catalog of Programs.' In: URL: http://www.bitsavers.org/pdf/ibm/share/SHARE_PgmCatalog_Jan77.pdf.

[11] Atsushi Akera (2001). 'Voluntarism and the Fruits of Collaboration: The IBM User Group, Share.' In: *Technology and Culture* 42.4, 710–736. ISSN: 0040-165X.

[12] Joanna Edson et al. (1956). 'SHARE Reference Manual for the IBM 704.' In: URL: https://latimesblogs.latimes.com/thedailymirror/files/share59.pdf.

imperative that majorities should not be "overbearing" to minorities; they should try to win them over through discussion rather than simply vote them down.

Cooperative users' groups became a popular model for distributed software development around other computers as well: the UNIVAC 1100 had USE, IBM's 4300 had GUIDE, Wang Equipment had SWAP, etc. The user groups were not just about exchanging knowledge and software between members, but often had a formal relationship with computer manufacturers, and they were able to exert considerable power over the business decisions these made (e.g., IBM on multiple occasion tried to kill the VM/CMS Operating System but SHARE intervened on behalf of its users[13]).

5.4 SOFTWARE AS A PACKAGE

Up until the end of the 1950s, software was generally considered an artefact without inherent commercial value, and there was little interest in claiming control or ownership over it. The prevailing idea at the time was that software by definition was inimical to standardisation and could never be sold as is: a general ledger or payroll software system would always need to be built specifically for the locale it would be used in.[14] Rather, manufacturers treated software as a way to boost the viability of their hardware, and they budgeted its development as part of their marketing costs. They freely shared the software they or their customers built through mail-order program libraries, recurring newsletter catalogues from which members could request a punched card or magnetic tape copy for a small fee. These software were considered general blueprints and "seldom if ever used as programs; rather, the overall system design was usually modified and then recoded to the particular user's requirements".[15]

The rising number of computer installations across the United States and the rapidly expanding capabilities of computers created a strong demand for more software, yet the shortage of workers with programming skills and the slow speed of program development made it difficult to meet. The inability of the in-house developed, custom-built model of software development to meet these demands resulted in the emergence of the software contracting industry. Not only did this industry increase the overall quantity of software in the US, but their business models – different from the manufacturer- or user-originated code – also created a new way of imagining and designing software in the 1960s: the software package.

For the organisations that could not find or afford in-house programmers, hiring development services from a contractor was an attractive way to create the programs they needed. Initially, these contractors would develop custom software based on the specifications worked out with their clients. Their business models were based on economies of scope: they specialised in building software for a very narrow market so they could reduce development costs by building up

[13] Ed Scannell (1984). 'IBM User Groups.' In: *Computerworld* 8.October.

[14] Arthur Norberg (1983). 'An Interview with Walter Bauer.' In: *Charles Babbage Institute*. URL: https://conservancy.umn.edu/bitstream/handle/11299/107108/oh061wb.pdf.

[15] Larry Welke (1980). 'The Origins of Software.' In: *Datamation* December, p. 127.

knowledge about the domain and a library of program code that could be used as templates.[16] Eventually, this repurposing of previously built software by contractors evolved into a growing interest for "pre-packaged" software across the computer industry. What the concept of the package aspired to do, was "make a program portable, transferable, and usable on a second computer unrelated to the first",[17] a paradigmatic shift in the way software was imagined[18]. Rather than serve as just a blueprint for building a new system, software was beginning to be considered as something that could be reused across locales.

The '62 CFO (Consolidated Functions Ordinary) package developed by IBM for the 1401 and released through their Application Program Library was an early attempt at such a package for life insurance companies, used for billing and accounting. The intent was that it could be "used broadly throughout the Life Insurance Industry as operational computer programs or as a guide in the development of personalized total systems on an individual company basis".[19] It was developed in consultation with representatives from several life insurance and, by 1964, used relatively widely (peaking at 200-300 installations), resulting in the creation of three CFO user groups by the insurance companies.

However, the market for software as a package never really took off, and at the end of 1960 was only about 10 percent the size of contract programming. There were a couple reasons for this stagnation. First, the belief that custom-developed software was inherently superior to pre-built solutions remained strong. Second, the lack of hardware standardisation made software developed for one computer generally incompatible for others, so the pool of potential customers for a package was small. And third, the bundling of software together with hardware by computer manufacturers meant third-party package developers were competing against the manufacturers' free software.

5.5 THE RISE OF THE SOFTWARE PRODUCT

A number of events in the mid to late 1960s made the market for software as a package more viable, and eventually culminated in the emergence of software as a product.

The problem of hardware standardisation was addressed by IBM's 1964 announcement of their System/360: a family of mainframes that would all be compatible with each other, making vertical and horizontal integration across market segments possible. For software contractors with a backlog of packages, this sig-

[16] Campbell-Kelly, *From airline reservations to Sonic the Hedgehog: a history of the software industry*.
[17] Welke, 'The Origins of Software.'
[18] The meaning of the term *software package* was not as clear-cut as presented here. For some, it referred to pre-made software developed by a manufacturer whose costs were bundled together with the hardware and available through their software library. Others use it to refer to just pre-made software in general, regardless of origin or delivery. At least one collection of oral histories reports that it was called a package because the client received, as a package, the program code, documentation, and some level of service (e.g., installation, maintenance) (Luanne Johnson [2002]. 'Creating the software industry-recollections of software company founders of the 1960s.' In: *IEEE Annals of the History of Computing* 24.1, pp. 14–42)
[19] JoAnne Yates (1995). 'Application Software for Insurance in the 1960s and Early 1970s.' In: *Business and Economic History*, pp. 123–134.

nificantly increased the pool of potential customers for each piece of software and thus its potential profitability.

The problem of manufacturers bundling their software with the hardware was similarly up to IBM to address, although other factors played a role. The company Applied Data Research (ADR) had released a package in 1966 called Autoflow, which produced automatic flowcharts of Assembly-based software. It had sold 300 copies by the end of 1968, which was relatively successful, but ADR believed their sales had been severely hurt by the fact that IBM had started to offer a free package called Flowcharter, which also produced software flowcharts (although not automatically and not based on analyzing the assembly code). ADR eventually brought a lawsuit against IBM, claiming that it was monopolising the software industry by bundling software for free with its devices. Whether the ADR lawsuit, the simultaneous anti-trust case brought by the US government over IBM's market dominance, or just the rising costs of software development and support that was "virtually bringing the company to its knees",[20] IBM announced that starting 1 January 1970, it would "unbundle" its software from its hardware and start charging separately for it.[21]

This unbundling did not necessarily create a market overnight by removing the competition of free software (many early packages did not have a direct IBM equivalent), but it did force companies to consider software costs when buying computers, allowing the notion of non-bundled software as a reasonable alternative to emerge. Before unbundling, buying software as a stand-alone package was an unusual decision that needed to be justified, often by the computer controller or the president of the company. Once IBM had normalised the idea, purchasing authority (for products up to a certain price range, e.g., US$15.000-20.000) moved down in the organisational hierarchy, making it easier to sell software packages.[22]

The idea that custom-built software was superior to packages never really went away, but the economic crunch of the 1970s made hiring programming contractors or maintaining an in-house development team simply too expensive for many companies. Packages, despite its design limitations, were significantly cheaper and could be operational within a matter of weeks instead of months, generating a faster return on investment. Rather than not use any software at all, organisations chose to buy a pre-made package and instead redesign their company structure to fit its design.[23]

With these three barriers removed, the software industry expanded significantly. In 1970 the annual turnover for all US American software firms was less

[20] Johnson, 'Creating the software industry-recollections of software company founders of the 1960s,' p. 37.

[21] Thomas Haigh (2002). 'Software in the 1960s as Concept, Service, and Product.' In: *IEEE Annals of the History of Computing* 24.1, pp. 5–13.

[22] Johnson, 'Creating the software industry-recollections of software company founders of the 1960s,' p. 27.

[23] Campbell-Kelly, *From airline reservations to Sonic the Hedgehog: a history of the software industry*, p. 98.

than US$0.5 billion; by 1979 this had quadrupled to roughly US$2 billion.[24] Software magazines were created to showcase all the different packages available and conference papers were written about how to evaluate and choose the best ones.

Slowly, the software package turned into a capital good: a software product. This commodification of software left a lasting imprint on its dominant designs. An early representative example of this transformation was Informatic's Mark IV, a file management system that generated programs for creating, maintaining, and reading data from punched cards, magnetic tapes, and direct access devices. First released in 1967, it grew out of tailor-made packages, before being turned into a stand-alone "product". John Postley, inventor of the previous Mark packages, argued that turning the system into a successful product required it to be bug free, robust, and fully supported by stand-alone manuals and documentation. This might seem trivial, but was a significant shift from the much more forgiving requirements when delivering software as a package of code and customer service. The "free" software package provided by the computer manufacturer "was supplied with few contractual obligations, and no matter how well the customer might be supported in practice, there was no legal requirement that the manufacturer supply a fully operational application".[25] The software *product*, on the other hand, was "a discrete software artifact that required little or no customization, either by the vendor or by the buyer; it was actively marketed, it was sold or leased to a computer user, and the vendor was contractually obligated to provide training, documentation, and after-sale service".[26]

With the business model change from software as a 'free' package to software as a tradeable product, there was a shift in responsibility over the quality of that software, and a subsequent appropriation of control over the software by its producers in order to better manage those responsibilities and ensure continued profit. Previously, software code was easy to access and trivial to replicate, which was economically acceptable because it had little to no value outside of the specific context it was built for. Software contractors were hired more for their development and support services than the program. Distribution and replication of software did not seriously endanger their business models. Now that more and more commercial value was contained in the program code, and it was possible to run it on multiple machines without requiring customisation, protecting the software code became imperative for the revenue streams of the company.

To protect an informational asset, legal instruments had to be invented or reappropriated. Patents were an ill-fit because they were designed for tangible objects rather than immaterial goods. Copyright law required a human-readable copy of the artefact to be submitted to the Copyright Office, something companies were obviously reluctant to do. In the case of Mark IV, Informatics decided to use the mechanism of trade secret laws, and required employees and customers to sign non-disclosure agreements. Most importantly, however, after roughly thirty years of mostly decentralised ownership of programs, the commodification of software

[24] Martin Campbell-Kelly (1995). 'Development and structure of the international software industry, 1950-1990.' In: *Business and economic history*, pp. 73–110, p. 74.

[25] Campbell-Kelly, *From airline reservations to Sonic the Hedgehog: a history of the software industry*, p. 99.

[26] Ibid., p. 99.

led developers to implement the first intentional technological barrier to a software's negotiability: they shipped their program in binary code, with the express purpose to limit the ability of their customers to access, read, and customise the software.

5.6 THE FIRST WAVE OF MICROCOMPUTERS

The microcomputer industry developed largely independently from the mainframe and minicomputer sector, but the evolution of the industry and the design of software follows an almost identical path. The first wave of microcomputers were sold as assemble-yourself kits for electronics hobbyists, containing circuit boards, wires, card connectors, a case, and instructions of often questionable quality. The first commercially successful microcomputer was the Altair 8800, announced in the now-iconic January 1975 edition of Popular Electronics (fig. 4). This was not a device with much practical applicability out of the box, but a new toy that provided an interesting challenge for hardware homebrewers and software hackers, captured perfectly in the headline of a later ad campaign: "Building your own computer won't be a piece of cake. (But, we'll make it a rewarding experience)".[27] The microcomputer was programmed by flipping switches, and the output came as flashing LED lights on the front. Using it for anything more substantial than that required buying external peripherals such as a keyboard, screen, disk drive, RAM extensions -– products which the company behind the Altair, MITS, sold to make a profit on the microcomputer.

Analogous to the early mainframes, these microcomputers did not come with any operating system or software programs, so users had to design their own. Initially using mostly machine language, until programing language interpreters were released that allowed users to write in something more human-friendly. Famously, the Altair 8800 gave Bill Gates and Paul Allen their first opportunity and first product, the BASIC interpreter called Altair BASIC. Unheard of at the time, Gates and Allen were paid for their software in royalties and received a small percentage of each sale made by MITS, rather than the more common licensing lump-sum. This business model would provide them with a steady influx of capital over the next years as they diversified into other programming languages (COBOL, FORTRAN) and other hardware.

Digital Research, founded by the married couple Gary Kildall and Dorth McEwen, was the first vendor of operating systems. Between 1974 and 1976, Gary Kildall designed CP/M – Control Program for Microcomputers – which could be run with minimal changes on any machine using the 8008 Intel microprocessor. At a time when a manufacturer could expect to spend US$50,000-100,000 on developing an operating system for a microcomputer, Kildall sold his CP/M by mail-order for US$75, and eventually licensed it (non-exclusively) for US$25,000 to IMSAI, the manufacturer of the Altair rival IMSAI 8080. Hugely popular, Digital Research adapted CP/M to run on other microprocessors as well, and by the end of the

[27] n.a. (1975). 'Building your own computer won't be a piece of cake.' In: *Popular Electronics* 4.

Figure 4. The Altair 8800 on the cover of the January 1975 edition of Popular Electronics magazine.

Figure 5. The welcome screen and interface of the Electric Pencil word-processing application created by Michael Shrayer in 1976.

decade could run on 200 different computers.[28] In much the same way that IBM's System/360 increased the market for software packages, CP/M's ad-hoc standardisation expanded the possible market for personal software: anything adapted to that operating system could run on a multitude of devices, liberating developers from having to significantly rewrite the software for each platform.

The application program market was still barebones in the mid-1970s, constrained by the limitations of the hardware and the fledgling state of programming languages and operating systems. One of the few commercially successful "practical" software products for the microcomputers at the time was a word processor. Michael Shrayer developed *Electric Pencil* out of personal necessity when he had to write the documentation for an assembler package he had created. State-of-the-art at the time, it showed the message "THE ELECTRIC PENCIL (C) 1977 MICHAEL SHRAYER" when the program was launched (fig. 5); nothing else would happen until the user started typing, which would replace the text.[29] Originally released in December 1976 for the Altair 8800, it become a runaway success and Shrayer quickly found himself rewriting the application for more than seventy other devices.

[28] Campbell-Kelly, *From airline reservations to Sonic the Hedgehog: a history of the software industry*, p. 206.
[29] Michael Shrayer (1977). *The Electric Pencil Word Processor: Operator's Manual*, p. 6.

The majority of programs available at the time were text-based games. One of the first and most widely used ones,[30] Star Trek, demonstrates the collective and negotiated character of these early programs. Originally written for a Sigma 7 minicomputer between 1971 and 1972 by a teenager called Mike Mayfield, Star Trek was a text-based strategy game inspired by the popular *Spacewar!* game. During the summer of 1972, a local Hewlett-Packard sales office allowed Mayfield access to their computer, and he rewrote it for their version of BASIC. It was subsequently added to the HP Contributed Program Library in February 1973 under the name STTR1. Other members of the library reprogrammed the game for their own hardware architectures, and added new game features along the way. Eventually, some of the many versions of the Star Trek game made it to the microcomputers. A copy of the game for the Altair 8800, written by Lynn Cochran, was featured as the cover story of the June 1976 edition of the SCCS INTERFACE magazine. The article included the entire software code (on just two and a half pages) and a block diagram explaining the general logic of the program (fig. 6). The renegotiability and shared ownership of the software was an explicit part of its identity. One article in the magazine lists the many different versions of the Star Trek game, and how one could embark on the "Enterprise" of modifying their own copy.[31] An ad in the magazine for yet another copy of the game highlighted that, in addition to "a complete program source listing", the package would also include "tips on how to 'patch' the program to add your own features".[32]

This collaborative and open design of software (games, at least) was the dominant imaginary, and the notion that programs should not be commodified held considerable sway among hobbyists. Such ideas clashed with those who wanted to extract value from their software. Writing angrily in February 1976, Gates accused members of the Homebrew Computer Club: "As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid? Is this fair?".[33]

5.7 THE GOLD RUSH OF APPLICATION PROGRAMS

The second wave of microcomputers started in 1977, kicked off by devices such as the Commodore PET (January), Apple II (April), and TRS-80 (August). These machines approached what we now recognise as a (personal) computer, at least in its hardware design: it hid away the electrical wiring behind a molded plastic case and came with a keyboard built in (fig. 7).

Although its exterior looked more approachable, significant technical skills were still required to overcome the lack of programs, poor documentation, and

[30] Art Childs (1976). 'Interfacial.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/, p. 2.

[31] Ralph Kiestadt (1976). 'Large Scale Systems: Software Choices for Star Trek.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/, p. 49.

[32] Inc. International Data Systems (1976). 'Patchable Star Trek/Space War Program Offered.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/, p. 41.

[33] Bill Gates et al. (1976). 'An open letter to hobbyists.' In: *Homebrew Computer Club Newsletter* 2.1, p. 2.

```
NULL 0
CLEAR 50
NEW


10 REM  ** STARTREK **   (3/14/76)
10 REM  A GAME OF INTRAGALACTIC WARFARE BASED ON NBC'S POPULAR TV SHOW
10 REM   ADAPTED FOR ALTAIR 8K BASIC (VERSION 3.1) BY L E COCHRAN
10 REM   AND REWRITTEN TO FIT (WITH 8K BASIC) WITHIN 12K OF MEMORY
10 REM    (EXPECT A 4 SEC PAUSE TO SET UP EACH QUADRANT, AND
10 REM      10 SEC AFTER "WORKING")


10 DIM D(5),K1(7),K2(7),K3(7),S(7,7),Q(7,7),D$(5)
20 D$=". EKB*"
30 D$(0)="WARP ENGINES"
40 D$(1)="SHORT RANGE SENSORS"
50 D$(2)="LONG RANGE SENSORS"
60 D$(3)="PHASERS"
70 D$(4)="PHOTON TORPEDOES":D$(5)="GALACTIC RECORDS"
80 INPUT"PLEASE ENTER A RANDOM NUMBER";E$:I=ASC(E$)
90 I=I-11*INT(I/11):FOR J=0 TO I:K=RND(1):NEXT:PRINT"WORKING-"
100 DEF FND(N)=SQR((K1(I)-S1)^2+(K2(I)-S2)^2)
110 GOSUB 610:GOSUB 450:Q1=X:Q2=Y:X=8:Y=1:X1=.2075:Y1=6.28:X2=3.28
120 Y2=1.8:A=.96:C=100:W=10:K9=0:B9=0:S9=400:T9=3451:GOTO 140
130 K=K+(NCX2)+(NCY2)+(NC.28)+(NC.08)+(NC.03)+(NC.01):K9=K9-K:GOTO 160
140 T0=3421:T=T0:E0=4000:E=E0:P0=10:P=P0:FOR I=0 TO 7
150 FOR J=0 TO 7:K=0:N=RND(Y):IF NCX1 THEN N=N*64:K=(NCY1)-Y:GOTO 130
160 B=(RND(Y)>A):B9=B9-B:Q(I,J)=K*C+B*W-INT(RND(Y)*X+Y):NEXT J,I
170 IF K9>(T9-T0) THEN T9=T0+K9
180 IF B9>0 THEN 200
190 GOSUB 450:Q(X,Y)=Q(X,Y)-10:B9=1
200 PRINT LEFT$("STARTREK ADAPTED BY L.E.COCHRAN 2/29/76",8):K0=K9
210 PRINT"OBJECTIVE:  DESTROY";K9;"KLINGON BATTLE CRUISERS IN";T9-T0;
220 PRINT"YEARS.":PRINT" THE NUMBER OF STARBASES IS";B9
230 A=0:IF Q1<0 OR Q1>7 OR Q2<0 OR Q2>7 THEN N=0:S=0:K=0:GOTO 250
240 N=ABS(Q(Q1,Q2)):Q(Q1,Q2)=N:S=N-INT(N/10)*10:K=INT(N/100)
250 B=INT(N/10-K*10):GOSUB 450:S1=X:S2=Y
260 FOR I=0 TO 7:FOR J=0 TO 7:S(I,J)=1:NEXT J,I:S(S1,S2)=2
270 FOR I=0 TO 7:K3(I)=0:X=8:IF I<K THEN GOSUB 460:S(X,Y)=3:K3(I)=S9
280 K1(I)=X:K2(I)=Y:NEXT:I=S
290 IF B>0 THEN GOSUB 460:S(X,Y)=4
300 IF I>0 THEN GOSUB 460:S(X,Y)=5:I=I-1:GOTO 300
310 GOSUB 550:IF A=0 THEN GOSUB 480
320 IF E<=0 THEN 1370
330 I=1:IF D(I)>0 THEN 620
340 FOR I=0 TO 7:FOR J=0 TO 7:PRINT MID$(Q$,S(I,J),1);" ";:NEXT J
350 PRINT"  ";:ON I GOTO 380,390,400,410,420,430,440
360 PRINT"YEARS =";T9-T
370 NEXT:GOTO 650
380 PRINT"STARDATE=";T:GOTO 370
390 PRINT"CONDITION:  ";C$:GOTO 370
400 PRINT"QUADRANT=";Q1+1;"-";Q2+1:GOTO 370
410 PRINT"SECTOR =";S1+1;"-";S2+1:GOTO 370
420 PRINT"ENERGY=";E:GOTO 370
430 PRINT D$(4);"=";P:GOTO 370
440 PRINT"KLINGONS LEFT=";K9:GOTO 370
450 X=INT(RND(1)*8):Y=INT(RND(1)*8):RETURN
460 GOSUB 450:IF S(X,Y)>1 THEN 460
470 RETURN
```



```
480 IF K<1 THEN RETURN
490 IF C$="DOCKED" THEN PRINT"STARBASE PROTECTS ENTERPRISE":RETURN
500 FOR I=0 TO 7:IF K3(I)<=0 THEN NEXT:RETURN
510 H=K3(I)*.4*RND(1):K3(I)=K3(I)-H:H=H/(FND(0)^.4):E=E-H
520 E$="ENTERPRISE FROM":N=E:GOSUB 530:NEXT:RETURN
530 PRINT H;"UNIT HIT ON ";E$;" SECTOR";K1(I)+1;"-";K2(I)+1;
540 PRINT" (";N;"LEFT)":RETURN
550 FOR I=S1-1 TO S1+1:FOR J=S2-1 TO S2+1
560 IF I<0 OR I>7 OR J<0 OR J>7 THEN 580
570 IF S(I,J)=4 THEN C$="DOCKED":E=E0:P=P0:GOSUB 610:RETURN
580 NEXT J,I:IF K>0 THEN C$="RED":RETURN
590 IF E<E0*.1 THEN C$="YELLOW":RETURN
600 C$="GREEN":RETURN
610 FOR N=0 TO 5:D(N)=0:NEXT:RETURN
620 PRINT D$(I);" DAMAGED.";
630 PRINT" ";D(I);"YEARS ESTIMATED FOR REPAIR.":PRINT
640 IF A=1 THEN RETURN
650 INPUT"COMMAND";A
660 IF A<1 OR A>6 THEN 680
670 ON A GOTO 710,310,1250,1140,690,1300
680 FOR I=0 TO 5:PRINT I+1;"= ";D$(I):NEXT:GOTO 650
690 IF D(4)>0 THEN PRINT"SPACE CRUD BLOCKING TUBES.";:I=4:GOTO 630
700 N=15:IF P<1 THEN PRINT"NO TORPEDOES LEFT":GOTO 650
710 IF A=5 THEN PRINT"TORPEDO ";
720 INPUT"COURSE (1-8.9)";C:IF C<1 THEN 650
730 IF C>=9 THEN 710
740 IF A=5 THEN P=P-1:PRINT"TRACK:";:GOTO 900
750 INPUT"WARP (0-12)";W:IF W<=0 OR W>12 THEN 710
760 IF W<=.2 OR D(0)<>0 THEN 780
770 I=0:PRINT D$(I);" DAMAGED, MAX IS .2 ";:GOSUB 630:GOTO 750
780 GOSUB 480:IF E<=0 THEN 1370
790 IF RND(1)>.25 THEN 870
800 X=INT(RND(1)*6):IF RND(1)>.5 THEN 830
810 D(X)=D(X)+INT(6-RND(1)*5):PRINT"**SPACE STORM,  ";
820 PRINT D$(X);" DAMAGED**":I=X:GOSUB 630:D(X)=D(X)+1:GOTO 870
830 FOR I=X TO 5:IF D(I)>0 THEN 860
840 NEXT
850 FOR I=0 TO X:IF D(I)<=0 THEN NEXT:GOTO 870
860 D(I)=.5:PRINT"**SPOCK USED A NEW REPAIR TECHNIQUE**"
870 FOR I=0 TO 5:IF D(I)=0 THEN 890
880 D(I)=D(I)-1:IF D(I)<=0 THEN D(I)=0:PRINT D$(I);" ARE FIXED!"
890 NEXT:N=INT(W*8):E=E-N-N+.5:T=T+1:S(S1,S2)=1
900 Y1=S1+.5:X1=S2+.5:IF T>T9 THEN 1370
910 Y=(C-1)*.785398:X=COS(Y):Y=-SIN(Y)
920 FOR I=1 TO N:Y1=Y1+Y:X1=X1+X:Y2=INT(Y1):X2=INT(X1)
930 IF X2<0 OR X2>7 OR Y2<0 OR Y2>7 THEN 1110
940 IF A=5 THEN PRINT Y2+1;"-";X2+1,
950 IF S(Y2,X2)=1 THEN NEXT:GOTO 1060
960 PRINT:IF A=1 THEN PRINT"BLOCKED BY ";
970 ON S(Y2,X2)-3 GOTO 1040,1020
980 PRINT"KLINGON";:IF A=1 THEN 1050
990 FOR I=0 TO 7:IF Y2<>K1(I) THEN 1010
1000 IF X2=K2(I) THEN K3(I)=0
1010 NEXT:K=K-1:K9=K9-1:GOTO 1070
1020 PRINT"STAR";:IF A=5 THEN S=S-1:GOTO 1070
1030 GOTO 1050:2L29E76C
1040 PRINT"STARBASE";:IF A=5 THEN B=2:GOTO 1070
1050 PRINT" AT SECTOR";Y2+1;"-";X2+1:Y2=INT(Y1-Y):X2=INT(X1-X)
1060 S1=Y2:S2=X2:S(S1,S2)=2:A=2:GOTO 310
1070 PRINT" DESTROYED!";:IF B=2 THEN B=0:PRINT".  .  .GOOD WORK!";
```



```
1080 PRINT:S(Y2,X2)=1:Q(Q1,Q2)=K*100+B*10+S:IF K9<1 THEN 1400
1090 GOSUB 480:IF E<=0 THEN 1370
1100 GOSUB 550:GOTO 650
1110 IF A=5 THEN PRINT"MISSED!":GOTO 1090
1120 Q1=INT(Q1+W*Y+(S1+.5)/8):Q2=INT(Q2+W*X+(S2+.5)/8)
1130 Q1=Q1-(Q1<0)+(Q1>7):Q2=Q2-(Q2<0)+(Q2>7):GOTO 230
1140 I=3:IF D(I)>0 THEN 620
1150 INPUT"PHASERS READY:  ENERGY UNITS TO FIRE";X:IF X<=0 THEN 650
1160 IF X>E THEN PRINT"ONLY GOT";E:GOTO 1150
1170 E=E-X:Y=K:FOR I=0 TO 7:IF K3(I)<=0 THEN 1230
1180 H=X/(Y*(FND(0)^.4)):K3(I)=K3(I)-H
1190 E$="KLINGON AT":N=K3(I):GOSUB 530
1200 IF K3(I)>0 THEN 1230
1210 PRINT"**KLINGON DESTROYED**"
1220 K=K-1:K9=K9-1:S(K1(I),K2(I))=1:Q(Q1,Q2)=Q(Q1,Q2)-100
1230 NEXT:IF K9<1 THEN 1400
1240 GOTO 1090
1250 I=2:IF D(I)>0 THEN 620
1260 PRINT D$(I);" FOR QUADRANT";Q1+1;"-";Q2+1
1270 FOR I=Q1-1 TO Q1+1:FOR J=Q2-1 TO Q2+1:PRINT"    ";
1280 IF I<0 OR I>7 OR J<0 OR J>7 THEN PRINT"***";:GOTO 1350
1290 Q(I,J)=ABS(Q(I,J)):GOTO 1340
1300 I=5:IF D(I)>0 THEN 620
1310 PRINT"CUMULATIVE GALACTIC MAP FOR STARDATE";T
1320 FOR I=0 TO 7:FOR J=0 TO 7:PRINT"  ";
1330 IF Q(I,J)<0 THEN PRINT"***";:GOTO 1350
1340 E$=STR$(Q(I,J)):E$="00"+MID$(E$,2):PRINT RIGHT$(E$,3);
1350 NEXT J:PRINT:NEXT I:GOTO 650
1360 PRINT:PRINT"IT IS STARDATE";T:RETURN
1370 GOSUB 1360:PRINT"THANKS TO YOUR BUNGLING, THE FEDERATION WILL BE"
1380 PRINT"CONQUERED BY THE REMAINING";K9;"KLINGON CRUISERS!"
1390 PRINT"YOU ARE DEMOTED TO CABIN BOY!":GOTO 1430
1400 GOSUB 1360:PRINT"THE FEDERATION HAS BEEN SAVED!"
1410 PRINT"YOU ARE PROMOTED TO ADMIRAL":PRINT K0;"KLINGONS IN";
1420 PRINT T-T0;"YEARS.  RATING=";INT(K0/(T-T0)*1000)
1430 INPUT"TRY AGAIN";E$:IF LEFT$(E$,1)="Y" THEN 110
```

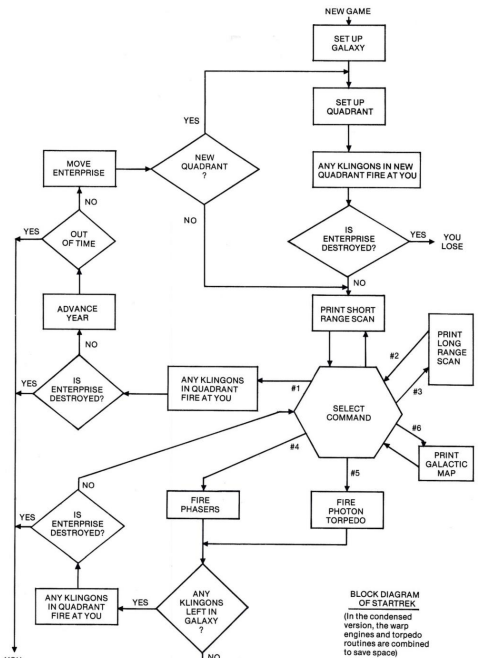



Figure 6. The program code and logic flowchart for the Star Trek game written by Lynn Cochran, published in the June 1976 edition of the SCCS INTERFACE magazine.

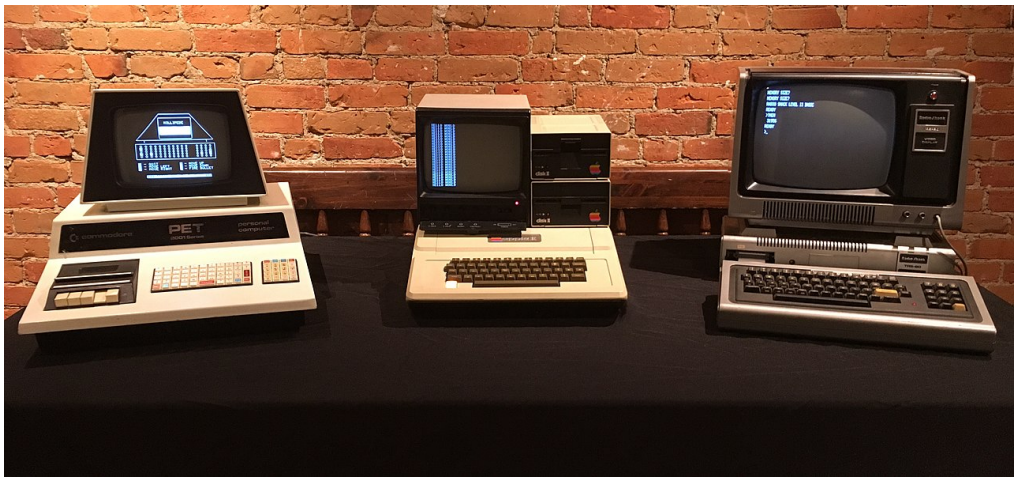
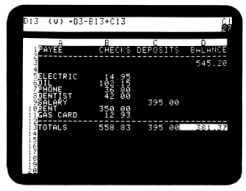


Figure 7. The second wave of commercial microcomputers. From left to right, the Commodore PET, Apple II, and TRS-80.

minimal support so the computer could be used as a tool rather than a toy. This changed at the end of the 70s, when the success of VisiCalc helped establish the application software market and usher in a four year gold rush that set the stage for the design and industry of software for years. Released initially for the Apple II in 1979 (and significantly buoying the platform's sales), VisiCalc was a spreadsheet program that would dynamically recalculate cell values in response to changes in other cells. Compared to other software available at the time, VisiCalc was exceptionally easy to use – an explicit design choice strongly influenced by the fact that its developers – Dan Bricklin and Bob Frankston – conceived of VisiCalc as "a product, not a program".[34] Whereas mainframe software relied on a low volume of sales for high prices (US$5,000 - US$250,000), the developers of VisiCalc decided to bet on a high volume-low price mass-market strategy instead, targeting everyday individuals instead of just large corporations with deep pockets. Supporting thousands of customers would be expensive, if not impossible, so Dan Bricklin and Bob Frankston relied on its design and documentation to be so simple and unsurprising that they did not have to have any further interaction with the customer after the point of sale: any feature that could not be concisely explained on the reference card was dropped (fig. 8). Priced initially at US$99, the company sold 1,000 copies per month on average. For four straight years, from 1979 through 1983, Personal Software/VisiCorp was the largest software company on the market.

Although games continued to be one of the main types of software, spreadsheets and similar "practical" applications such as word processors, database managers, and business graphics became multimillion dollar categories in their own rights. The professional appeal of the microcomputer was further solidified when IBM announced it would enter the business as well, and released the IBM PC in 1981. Big Blue's reliable reputation helped it quickly capture a large market share, but more important for the software industry was that the IBM PC had an

[34] Robert M. Frankston (2015). 'Implementing VisiCalc.' In: URL: https://rmf.vc/implementingvisicalc.



VISICALC™

A Visible Calculator

For the

APPLE II

REFERENCE CARD

A Product of

Software Arts, Inc.

Distributed Exclusively By

PERSONAL SOFTWARE INC.

592 Weddell Drive
Sunnyvale, CA 94086
(408) 745-7841

9-79 V1.35

MOVING THE CURSOR

←→	Moves the cursor left, right, up or down.
space bar	Switches the direction indicator between horizontal (–) and vertical (!).
:	If two windows, moves the cursor from one window to the other.
>	Go To command. Type the coordinates of the entry where you want the cursor to go; end with RETURN.

THE ESC KEY

The ESC key is used to recover from simple typing mistakes. It usually erases the last thing that you typed. If you press ESC enough times, it will abort what you are doing and return VisiCalc to a blank prompt line.

SETTING A LABEL ENTRY

Label entries start with a letter (A–Z), or with the quote character ("). Terminate entering a label entry by pressing ←, →, or RETURN. Correct errors by pressing ESC. The prompt line will say LABEL while a label entry is being typed.

SETTING A VALUE ENTRY

A value entry displays the calculated value of the expression stored at the entry. Expressions consist of numbers, coordinates of other value entries (value references), functions (such as @SUM), arithmetic operators (+ – * / ^) and/or parentheses. Expressions are evaluated strictly from left to right except as modified by parentheses. You must start an expression with a +, a digit (Ø–9), or one of the symbols @ – (. or #. The prompt line will say VALUE while an expression is being typed. Terminate entering an expression by pressing ←, →, or RETURN. Errors can be corrected by pressing the ESC key. Examples of expressions are:

12.34	A normal number
.1234E2	A number in scientific notation
2+2	An arithmetic expression
+B4	A value reference
2*B4	An expression with a value reference
2*(3+4)	An expression with parentheses

If you press ! while entering an expression, VisiCalc will calculate the value of the expression so far and replace the expression on the edit line with the number which results from the calculation.

VALUE REFERENCES

An expression at one entry can refer to the value of another entry, and the value of such an expression can be automatically recalculated when the value of the other entry changes. Value references are allowed in expressions wherever numbers are allowed. A value reference is made by either typing the coordinate of the desired entry (such as B5), or by "pointing" to the entry with the cursor (in this case, the coordinate will be "typed" automatically by VisiCalc). If an expression starts with a value reference, it must be preceded by a + character.

In order to insert the current value of another entry into an expression as a number, which will be unaffected by later changes to the other entry, type a value reference followed by the character # (e.g. B5#). If # is used by itself, it will be replaced by the current value of the expression stored in the entry you are changing.

FUNCTIONS

@SUM(*list*)	Calculates the sum of the values in *list*. See **LISTS**, below.
@MIN(*list*)	Calculates the minimum value in *list*.
@MAX(*list*)	Calculates the maximum value in *list*.
@COUNT(*list*)	Results in the number of non-blank entries in *list*. Maximum number of entries in the list is 255.
@AVERAGE(*list*)	Calculates the average of the non-blank values in *list*. Maximum number of entries in the list is 255.
@NPV(*dr*,*range*)	Calculates the net present value of the cash flows in *range*, discounted at the rate specified by expression *dr*. The first entry in the range is the cash flow at the end of the first period, the second entry is the cash flow at the end of the second period, etc. See **ENTRY RANGES**, below.

@LOOKUP(*v*,*range*)	Compares the value *v* to the values of successive entries in *range*, and selects a corresponding value from the column or row immediately to the right or below the entries in *range*, as the result of the function. The values in *range* are normally in ascending order, and the result is the value corresponding to the last entry in *range* that is less than or equal to *v* before an entry greater than *v* is found. If the first entry in *range* is greater than *v*, the result of the function is NA.
@NA	Results in a "Not Available" value that makes all expressions using the value display as NA.
@ERROR	Results in an "Error" value that makes all expressions using the value display as ERROR.
@PI	Results in 3.1415926536.
@ABS(*v*)	Results in the absolute value of *v*.
@INT(*v*)	Results in the integer portion of *v*.
@EXP(*v*) @SQRT(*v*) @LN(*v*) @LOG1Ø(*v*) @SIN(*v*) @ASIN(*v*) @COS(*v*) @ACOS(*v*) @TAN(*v*) @ATAN(*v*)	Calculates the appropriate function. The trigonometric calculations are done in radians.

EXAMPLES OF FUNCTIONS

```
@SUM(B4 . . . B15)
@MIN(1Ø@, F4 . . .F11,@SUM(B4 . . .B15) )
@MAX(@,F4-F5)
@NPV(.15,B4 . . .F4)
```

ENTRY RANGES

An entry range consists of a number of entries that are next to each other in a row or column, such as B2, B3, and B4, or B2, C2, D2, and E2. You enter an entry range by specifying the coordinate of the first entry in the range, then typing an ellipsis (. . .—you need only type the first period, VisiCalc will fill in the others), and then specifying the last entry. For example, the entry ranges just mentioned would be B2 . . . B4 and B2 . . . E2, respectively. Coordinates are specified by either typing them, or "pointing" to the desired entry with the cursor.

LISTS

A list consists of a series of expressions and ranges separated by commas. See the examples of lists in **EXAMPLES OF FUNCTIONS**, above.

COMMANDS

/B	Sets an entry to blank. Doesn't take effect unless you follow it with ←,→, or RETURN. Does not affect /F formats set at the entry.
/C	Clears the screen, sets all entries to blank, resets formats, windows, titles, etc. VisiCalc will wait for you to type a Y to confirm that you indeed want to erase all entries.
/D	Deletes the row (/DR) or column (/DC) on which the cursor lies.
/F	Sets the display format of an entry to one of the following formats: general (/FG), integer (/FI), dollars and cents (/F$), left or right justified (/FL or /FR), or graph (/F*).
/FD	Resets an entry to use the global default format instead of an explicit format set with a /F command.
/GC	Sets the column width. Requests a number from 3 to 37; end with RETURN. The column width can be changed on a per window basis.
/GF	Sets the global default format that determines the display format of all entries without explicit format settings set with a /F command. Requests one of the same display formats used by the /F command. The global default format is a per window setting.
/GO	Sets the order of recalculation to be down the columns starting at entry A1 (/GOC), or across the rows starting at entry A1 (/GOR).
/GR	Sets recalculation to be automatic (/GRA) or manual (/GRM). You can always cause a manual recalculation of all entries by pressing the ! key.

/I	Inserts a row (/IR) or column (/IC) just above or to the left of the row or column on which the cursor lies.
/M	Moves an entire row or column to a new position. Prompts you to move the cursor from the row or column which you want to move to the destination row or column just before which the entries moved should reappear. End with RETURN.
/P	Print command. See **PRINTING**, below.
/R	Replicate command. See **REPLICATE**, below.
/S	Storage command. See **STORAGE COMMANDS**, below.
/T	Sets a horizontal title area (/TH), a vertical title area (/TV), sets both a horizontal and vertical title area (/TB), or resets the window to have no title areas (/TN).
/V	Displays VisiCalc's version number on the prompt line. The version number will disappear as soon as you type something else.
/W	Window control. Splits the screen into two windows at the cursor position (/WH for horizontal, /WV for vertical), or returns the screen to one window (/W1). Windows may be synchronized (/WS), or returned to unsynchronized (/WU).
/–	Repeating label. Requests the contents of a label entry; end with ←,→, or RETURN. The contents of the label will be repeated over and over to fill the entry, no matter what the column width.

PRINTING

The /P command lets you output to the printer.

1) Position the cursor at the upper left corner of the rectangle of entries that you wish to print and type /P.

2) VisiCalc will prompt for the slot number of a printer or communications card; type the number followed by RETURN. If you press just RETURN, it will scan the slots and automatically pick one that has an appropriate device.

3) Lines printed by VisiCalc are usually terminated by the character pair RETURN/LINE FEED. To suppress the LINE FEED, type a minus (–) at this point. If you press + at this point, a RETURN will be output immediately.

4) If you want to output setup characters to the printer, type ", then type the characters, and end with RETURN.

5) Move the cursor to (or type the coordinates of) the lower right corner of the rectangle of entries to be printed out, and press RETURN.

Note: the screen may be overwritten by the printer controller. It will be returned to normal when the /P command is finished. You may stop printing at any time by pressing CTRL-C.

If, in step 2, you type D, VisiCalc will prompt you for a file name to receive the output as a text file. You respond in the same manner as for the /SS command (see **STORAGE COMMANDS**, below).

REPLICATE

The /R command allows you to make copies of entries.

1) Position the cursor on the first entry that you wish to replicate, and type /R.

2) VisiCalc will ask for the coordinates of the source (what you want to replicate). If you are just replicating the current entry, press RETURN. If you want to replicate a range of entries, type an ellipsis and provide a coordinate to complete an entry range specifying the source, ending with RETURN.

3) VisiCalc will ask for the coordinates of the target (where you want the copies to go). This may be a single coordinate or an entry range. End the target with RETURN. If you are replicating a source range of entries, the first source entry will be replicated into the target range, and succeeding source entries will be replicated into correspondingly succeeding target ranges.

4) If the expression being replicated contains value references VisiCalc will ask you, for each value reference, whether it should not be modified (respond by typing N), or should always refer to the entry in the same relative position (type R).

STORAGE COMMANDS

The /S command lets you save and load the current entries, using either diskette or cassette.

When using diskette, VisiCalc will prompt for a file name. You may either type a file name (with optional drive number and slot number, ended with RETURN), or just press RETURN (with a blank file name, and optional drive or slot specification such ", D2"). If you don't provide a file name, VisiCalc will display the name of the first text file on the diskette. If that is the file name that you want to use, press RETURN; otherwise press → and VisiCalc will show you the name of each successive text file on the diskette. When you have found the file name that you want, press RETURN. You may edit the file name before executing the command by typing additional characters to add to the name, and/or using the ESC key to erase part of it.

When using the cassette, VisiCalc will ask you to ready the recorder before it executes the command; press RETURN when ready.

The storage commands can be aborted by pressing CTRL-C.

/SS	Save all entries, titles, and window settings in a file on the diskette.
/SW	Save all entries, titles, and window settings on a cassette.
/SL	Load the contents of all entries that were saved in a file on the diskette. This command does not blank out all entries before doing the load; if that is desired, use the /C command first.
/SR	Load the contents of all entries that were saved on a cassette. This command does not blank out all entries before doing the load; if that is desired, use the /C command first.
/SD	Deletes the specified file on the diskette. Type Y to confirm.
/SI	Initializes, or formats, a blank diskette. Existing contents, if any, of the diskette will be erased. Displays default slot and drive numbers on the edit line. Edit, if desired, as described above for file names. Press RETURN to start initialization.

Figure 8. The VisiCalc reference card, minus the last page which showed an annotated screenshot of the interface. The reference card was shipped in a brown, fake leather folder together with 5 1/4" diskettes, a manual, and a registration card. See http://www.bricklin.com/history/saiproduct1.htm for more detail.

open hardware architecture (in contrast with the proprietary Apple II system, for example), making it possible for third-parties to develop software and hardware for the platform. The IBM hardware design became a de-facto standard, reducing (but not eliminating) the technical and business challenges of device-specific software development, and broadening the software market. The lack of practical software had kept the microcomputer a hobbyist novelty, but these applications and the legitimisation by IBM helped reimagine the microcomputer as an essential business tool for white-collar workers and a household economics device for non-technical users.

In addition to applications themselves, there also emerged a lively after-market software industry (referred to as add-ons, add-ins, plug-ins, accessories). Applications were often not just a tool for executing the pre-programmed functions, but also a development environment in their own right, or at the very least exposed enough of their code to make it possible for other code to interoperate with it. George Tate, CEO of Ahston-Tate, the company behind the immensely popular database application dBASE, explained how they "never really sold dBASE as just an end-user product. [...] It's always been – always and forever – a product that serves two markets". One was to use the database as a database, another was for skilled users to turn their workflow into a program "and have something that his secretary could use, or have something he could market".[35] The "accessories" built around dBASE fell into two categories: first, utilities to enhance the parent software (e.g., Quickcode by Fox&Geller helped with the creation of databases); second, "vertical market" packages, which added functionality to dBASE that served a particular niche community (e.g., Abstrat by Anderson-Bell added statistical analyses for accounting). Many of the other dominant software packages had and supported similar extensibility, allowing the software to be renegotiated by its users through third-party packages or programming their own customisations.

The particular way microcomputer applications were commodified reconfigured the means and relations of software production, and thus the potential for negotiating its design. Where software contractors and manufacturers relied on economies of scope, application programs for the microcomputer were based on economies of scale. This mass-marketisation intentionally disconnected the software's user from the software's developer, closing off one way people would traditionally customise the software. Because early consumers of applications were mostly technical hobbyists, the software's designs did allow for reprogramming, either by making the source code accessible or supporting the development of new functionality from within the application itself. However, because mass-marketisation also expanded the target market for software from niche home-brewers to everyday users, the proportion of consumers who had the skills (or interest) to reprogram their software themselves decreased significantly. Without the competences or access to more capable peers, the negotiation of a software's design was now limited to whatever add-ons and plugins were available to the user.

[35] Scott Mace (1983). 'Software accessories enhance software programs.' In: *InfoWorld* 5.10, p. 24.

5.8 THE SEARCH FOR SOFTWARE INTEGRATION

After the gold rush era of application software, the rest of the 1980s saw the software industry mature. By 1983, a handful of applications had established themselves as the market leaders and barriers to entry made it harder for small start-ups to compete. The increased technical capabilities of microcomputers meant software development became more complex and expensive, the human-computer interaction expertise necessary to build easy-to-use software was mostly locked inside existing companies, and an expensive advertising blitz was required to secure a limited spot on the shelf in the computer store (an estimated 35,000 products were competing for the 200 slots).[36]

In addition to these socioeconomic factors, there was also a simple technological bottleneck on the growth of the personal software market: it was practically impossible to use more than one application at a time. First, switching software would generally take several minutes, as the user had to close the current application, remove the storage disk, load the second disk, and start up the new program. Second, the lack of data standardisation meant that transferring data between application was generally impossible, unless you were lucky and the software application had made their format public and was popular enough that others built compatibility with it. Third, applications all had their own set of commands that had to be memorised. Understanding the basic functions of a piece of software could take weeks, and mastering all of them even longer. Having multiple applications use the same commands for different outcomes made it practically unrealistic for a user to learn more than two or three programs.

"Software integration" became the holy grail for applications in the 1980s. Its aim was to let the consumer use multiple programs side-by-side, easily share data between them, and unify the commands and interface so that skills would be transferable. This drive towards integration was the main impetus for most software innovation during the 1980s and helped application design converge into what we know today. There were three main designs the industry explored: application families, integrated packages, and windowed application managers.

5.8.1 *Application families*

Application families often started after a software developer achieved some success with one of the four main software products – word processors, spreadsheets, database managers, and graphing tools – and then tried to branch out to the other categories. These applications were purchased separately, and "integration" meant they had a shared command structure and data format. The main disadvantage of this "family" approach was that a user had to buy all the different applications to benefit from the integration, and while the company's spreadsheet might be the industry leader, there was no guarantee that their word processor would be better than (or ever comparable to) other applications on the market.

[36] Campbell-Kelly, 'Development and structure of the international software industry, 1950-1990,' p. 97.

Data format compatability was not reserved just for applications of the same family. Bob Frankston, one of the developers behind the VisiCalc application, created the text-based Data Interchange Format (DIF) so that VisiCalc could exchange data with VisiPlot, their graphing application. By virtue of VisiCalc's market dominance, DIF became an ad-hoc standard because other software companies tried to use read/write compatability with VisiCalc to claim a share of the market. This network effect also worked in favour of the dominant application, so companies would often build explicit compatability support into their data format. For example, the files of Ashton-Tate's dBASE II -– one of the best-selling database programs -– began with 500 characters listing the names, types, and sizes of all database fields, followed by the data itself, which made it possible for developers of other BASIC programs to build around it. WordStar, one of the largest word processor packages released in 1979, stored its data in a plain text file, making reading and linking to it even easier.

While these approaches made data integration technologically possible, experientially there was still a big disconnect between applications, because a user would still have to quit one program and load another to work with that data.

5.8.2 *Integrated packages*

Integrated application packages similarly tried to make it possible for customers to combine the most commonly used productivity software, but rather than selling these as separate applications, the idea was to build "one-application-to-rule-them-all". ContextMBA (1981) was the first of these integrated application packages, although it preceded the term. It combined a spreadsheet, database manager, word processor, form creator, graphics tool, and "telecommunications" functionality (i.e., using phone lines to transfer data between the application and a mainframes computer, other ContextMBA users, or a timeshare system). Each of these were referred to as a "context" and a user could create live links between data entered in any of them, so if a spreadsheet was updated, so were the graphs created based on that data.

Since there was still no standardised microcomputer architecture, ContextMBA was written in Pascal, a higher-level language that could be run on any device with a Pascal interpreter, but which also made it slower than if it was written for a specific architecutre in assembly or machine language. It proved too slow for most users. Reviews in PC Magazine of June 1983 summarised that ContextMBA "sacrifices efficiency for integration"[37] and concluded that "[t]he idea of an integrated management system is excellent. [...] Nevertheless, a good set of compatible text editing, database, graphics, and spreadsheet programs that runs under MS-DOS is probably a better investment".[38]

Lotus 1-2-3 was the second integrated package that was released, and turned out to be an overnight success (fig. 9). Launched in October 1982 and shipped in January 1983, Lotus 1-2-3 was the brainchild of Mitch Kapor, former head of

[37] Mark S. Zachmann (1983). 'Context MBA: Half a Step in the Right Direction.' In: *PC Magazine* 2.1, p. 123.
[38] Ibid., p. 131.

development at VisiCorp and creator of the popular VisiPlot and VisiTrend accessory applications for VisiCalc. The software was more limited than ContextMBA and arguably only barely qualifies as an 'integrated' package: it was mostly a spreadsheet program with additional database, word processing, and graphics capabilities. It was integrated in the sense that all these components were usable in the spreadsheet and shared the same base commands, but they were not fully-fledged applications. Data from VisiCalc, dBase II, and WordStar could also be added after being "analysed" by Lotus.

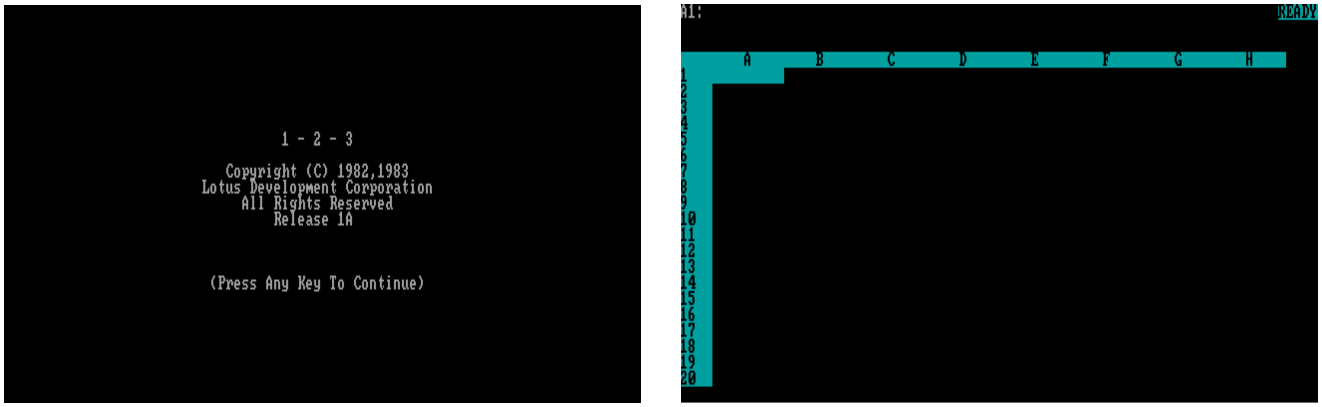


Figure 9. The splash screen and spreadsheet view of Lotus 1-2-3 release 1a (1983).[39]

One of Lotus' most popular features was that it allowed users to write macros – sequences of keystroke commands that could automate part of the user's interaction with the software and system and thus create quite complex programs (e.g., parsing comma-delimited files). This generated a very active after-market industry of macro packages (in v1.0) and add-ins (in v2.0) (fig. 10). These were initially seen as parasitic by Lotus, but were embraced when it became clear that the grassroots development of additional functionality boosted its popularity in niche communities and helped fortify its market position. It needed this, because Lotus never successfully diversified its offerings. Nearly all of its other software products were commercial failures.

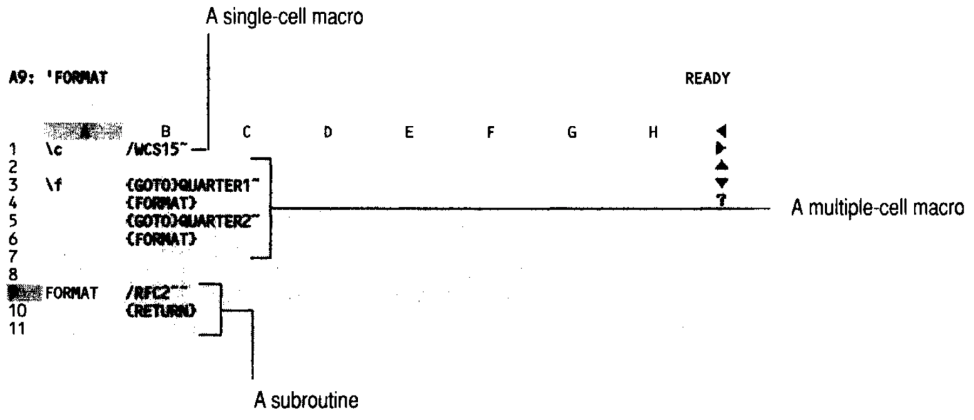


Figure 10. An example macro of Lotus 1-2-3 release 2.3 (1991).[40]

When version 2.0 of Lotus was released in September 1985, little new was added beyond supporting add-on development by exposing how other software could integrated with Lotus. Lotus Corporation also created considerable institutional support for add-on developers. They published all the file formats of its products, set up the Development Services Department to provide telephone support, organised conferences (the first one was attended by 400 people), and created the Registered Developer Program. Before these changes, insider knowledge or reverse engineering of Lotus was required to build an add-on, but with explicit documentation and support, the numbers of third-party software exploded. In 1987, Lotus magazine reported that over 1.000 add-ons were created.

Although Lotus' extensive support for add-on development made its design explicitly negotiable, the company fought heavily to retain control in other ways, with ramifications for negotiable designs in the software industry at large. Competitors to Lotus often tried to synchronise their menu tree structure with the one from Lotus. This was helpful to users because they did not have to learn how to use a new interface, but it also made it possible for the macros that they or their organisation had written – which could reach thousands of commands and represented a significant investment – to be interoperable with the other software's menu as well. Lotus Development Corp responded to these designs that would allow users to have more control over their software environment by filing copyright infringement lawsuits. Their first victim was Paperback Software International in 1987, which had developed a spreadsheet that looked identical to Lotus 1-2-3. At the time, the legal system only recognised source code as something that could be copyrighted, so it required a new interpretation of copyright to also include what became known as the "look and feel" of a piece of software. This lawsuit was a widely publicised event because it was believed that it could reshape the vision of what the software industry was working towards. One attorney observed in the magazine InfoWorld that "[i]f Lotus wins these lawsuits, it ... may kill all hope of there someday being a standard user interface across all software packages".[41]

Lotus Development Corp won the lawsuit in 1990, and it quickly went on to also sue Borland International, developer of the competitor application Quattro Pro. Quattro Pro had its own unique interface, but also a mode that looked the same as Lotus 1-2-3, and thus allowed macros developed in one to work in the other. The lawsuit dragged on for six years, with multiple conflicting verdicts at different levels in the US court structure, before eventually reaching the Supreme Court where it ended in a 4-4 stalemate between the judges. By default, this meant an earlier judgement in favour of Borland would stand, and menu hierarchies were deemed uncopyrightable. However, the six years of uncertainty for developers had had a chilling effect on technical interoperability and visual synchronicity as something that should be pursued in software design (in particular because the first verdict against Borland happened in Massachusetts, where a lot of software developers had their offices). Without these qualities, the freedom to easily switch between applications without having to abandon the skills and data developed in

[41] Doug Derwin (1987). 'Overheard...' In: *InfoWorld* 9.4, p. 35.

Figure 11. Lotus' Symphony (left) and Ashton-Tate's Framework. Note the explicit revocation of user ownership and control in Framework's splash screen: "You do not become the owner of the package nor do you have the right to copy or alter the software".

another was significantly constrained, and instead users were becoming walled-off as soon as they made their first choice.

The integrated packages industry reached its peak in 1984 with the simultaneous launch of Lotus Development Corps' Symphony and Ashton-Tate's Framework (fig. 11). Symphony was meant to be the full-blown integrated package that 1-2-3 aspired to be and included the classic combination of spreadsheet, word processor, graphics, database, and communications. Users could switch between each of these environments by pressing ALT+F10 and see a different view of the same underlying data. Ashton-Tate's Framework included the same applications, and organised them using the metaphor alluded to in its name. Each application was launched inside a frame, and the user could have multiple frames open at the same time, providing visual integration. Frames could contain other frames, data could be linked between frames such that changes would propagate, or frames could target other frames as output containers.

Integrated packages never delivered on their promises and were mostly commercial failures. The size of the software meant that almost all of them required an additional external hard disk to store the programs – or instead continuously swap between a fistful of floppy disks – and made them much slower than other applications. Even though the command structures were integrated, the sheer number of commands available made these software synonymous with difficult

to use in popular media.[42] The quality of, for example, the spreadsheet and the word processor part of the package could vary considerably, so while users paid a premium price for the integrated packaged they also have to buy other stand-alone applications. Data transfer also fell short of the claims made by vendors, with many integrated packages offering just cut-and-paste techniques for sharing data between different programs, rather than transclusion and live links.[43]

5.8.3 *Windowed Application Managers*

In addition to the multi-purpose application approach to integrated software, software developers also explored application managers that could show multiple, unrelated programs side-by-side and allow users to quickly switch between them: the window environment These efforts were not separate from the large integrated package approach, but happened in conjunction with them, before eventually being pitted against each other in the press.

The 1981 Xerox Star was the first commercial product that had a *graphical* window environment. The user accessed the software through icons and executed operations through menus. It also introduced the mouse as the main way to interact with those elements. This helped reduce the need for unified or standardised command structures that was one of the driving forces behind full package integration, because now users could see the different possible operations and click on them, rather than having to memorise the key combination that would execute it.

The 1983 Apple Lisa – on paper an acronym for Locally Integrated Software Architecture, but in reality named after the child Steve Jobs refused to acknowledge – is credited for popularising the graphical window manager (fig. 12). The Lisa was never a commercial success for a number of reasons, including its hefty price-tag, internal conflicts at Apple, and the fact that the more limited but cheaper Macintosh was released a year later. On the Lisa, the user could see multiple applications in the integrated package side-by-side and quickly switch between them (but not use them at the same time) through overlapping windows. This window environment was tied together with the integrated packaged called the Lisa Office System, which included LisaWrite, LisaCalc, LisaDraw, LisaGraph, LisaProject, LisaList, and LisaTerminal. To transfer data between these different applications, the Lisa used cut-copy-and-paste commands, and introduced the term 'clipboard' to refer to the temporary storage where the data would be kept in the process. Compared to the live links between data that other software offered, this model of data integration was considered to be on the most limited side of the spectrum.

Other dominant software companies quickly followed with their own windowed application managers: IBM's TopView, QuarterDeck's DESQView, Digital Research's Graphics Environment Manager, VisiCorp's VisiOn, and Microsoft's Windows (fig. 13). Some came with integrated packages, others were simply frameworks for third party software. By virtue of being the leading company for microcomputer

[42] Christine McGeever (1984). 'A Look at Lotus for the Mac.' In: *InfoWorld* 6.47.
[43] Paul Korzeniowksi (1984). 'Multi-application Packages: Who Needs Them?' In: *InfoWorld* 18.33.

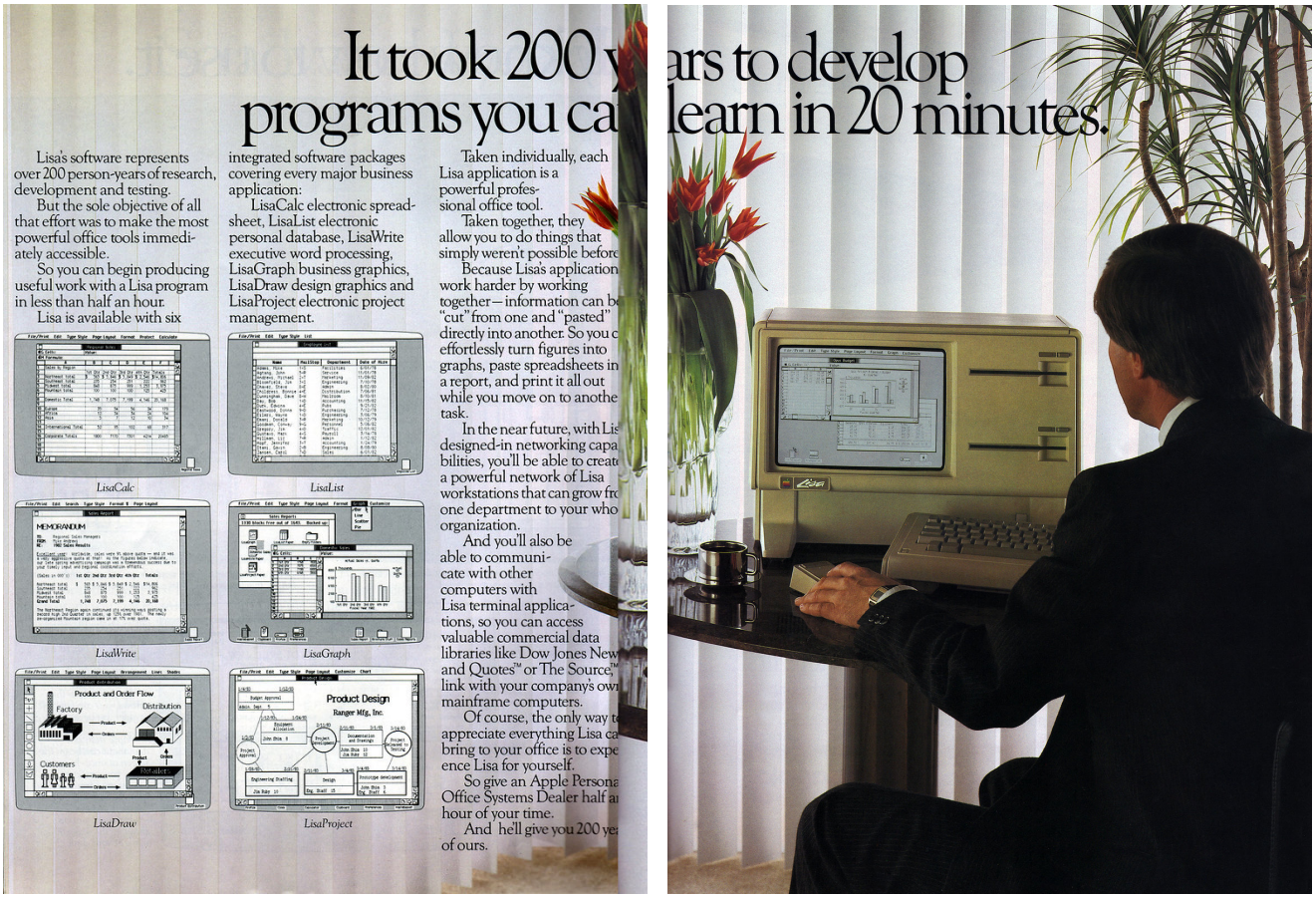
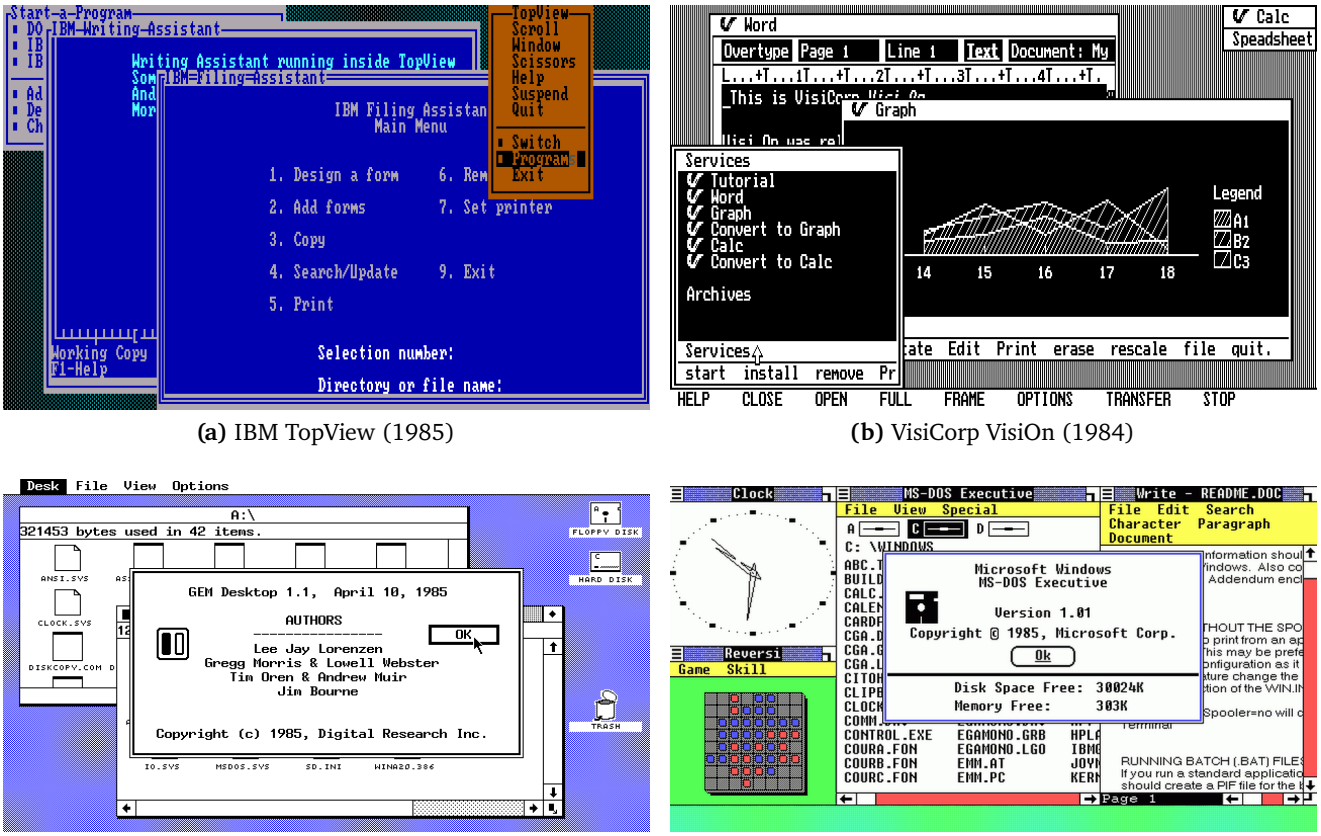


Figure 12. Advertisement for the Apple Lisa in Personal Computing 1983.[44] Note the promotion of the Lisa Office programs and the cut and paste functionality.

software in revenue, VisiCorp's product was the most highly anticipated. It turned out to be a fatal product for the company. VisiCorp had been working on their version since 1982 and committed significant resources, and when it was finally launched in January 1984 for the IBM PC it got positive reviews but made hardly any sales. The minimum hardware requirements were too high for most users and even a price cut from US$495 to US$95 within the first month did not save it. Another factor was that Microsoft had undercut VisiCorp by announcing in November 1983 that it would release their window-based environment the next spring. It chilled the interest for VisiOn – something better might be just around the corner – but Microsoft was overconfident and after months of technical difficulties only released their product – Windows 1.0 – in January 1985.

The "window wars" of 1984-1985 were the most exciting event in the software industry at the time. Integrated packages and windowed environments were two paths towards the same goal – functional multitasking — and were constantly pitted against each other in the press. Some industry observers proclaimed that the window environment would be the death of integrated software packages, while



(a) IBM TopView (1985)

(b) VisiCorp VisiOn (1984)

(c) Digital Research Graphic Environment Manager (1985)

(d) Microsoft Windows (1985)

Figure 13. Windowed Application Managers released between 1984-1985.

others believed that they could co-exist.[45] Like the integrated packages, however, all early window environments were market failures. The software required too much from the hardware at the time and third-party applications needed to be rewritten from scratch to take advantage of the added benefits, which few were keen to do. The casualties were steep. VisiCorp and Digital Research had to fire half of its employees. Eventually, VisiCorp was sold off to Paladin Software, and Digital Research CEO Gary Kildall (of CP/M fame) resigned.[46] Only IBM and Microsoft had enough capital to overcome the financial blow of these failures.

5.8.4 *Component Software*

Component software was seen as the answer to the failures of integrated packages and windowed application managers: a hybrid model which would allow customers to use multiple applications from different vendors side-by-side, while still getting the data compatibility and functional interoperation from integrated packages. Writing about the battle between integrated software and windowing

[45] Winn L. Rosch (1985). 'Can Integrated Software Co-Exist with Windows?' In: *PC Magazine* 4.2, p. 60.

[46] Campbell-Kelly, *From airline reservations to Sonic the Hedgehog: a history of the software industry*, p. 250.

environments, one journalist predicted that "[s]oftware will become like stereo equipment – the low end will be integrated and the high end will be components".[47] In the late 1980s and early 1990s, it was touted as the next paradigm shift in computing. Apple confidently wrote: "In the 1980s, the graphical user interface revolutionized personal computing, enabling big leaps in user productivity and ultimately making obsolete all the applications standards of the day. In the 1990s, Apple believes the next major software revolution will be component software".[48] Compound documents would be the resulting artefact, "something like a display desktop that can contain visual and information objects of all kinds"[49] including text, graphics, spreadsheets, calendars, video, buttons, a newsreel, etc. The goal of this paradigm, stemming from the philosophy of object oriented programming, was to decenter the application as the container of commands and programs, and instead have the document take centre stage, with the data editable in place through contextual menus. For software developers, it was projected that they would move away from building fully-fledged applications and instead a lively component market would appear.[50]

Still developing and promoting its graphical window environment, Microsoft was an early proponent of this software vision and released the Dynamic Data Exchange (DDE) standard in 1987 as part of Windows 2.0, which "lets users dynamically link applications, automating tasks performed between multiple programs".[51] The first commercial software to demonstrate this ability was Microsoft Excel, also launched in 1987, but it took another two years for third-party applications to appear that took advantage of it. For example, Autocad could connect its diagrams to Microsoft Excel, such that changes in the spreadsheet would also update the graphics. Microsoft pushed these ideas further in 1990 and released their Object Linking and Embedding (OLE) protocol together with Windows 3.0, the first of their windowed application managers that was actually successful. Where DDE just propagated plain text messages between applications, OLE could maintain active links between data across programs and fully embed the content. This allowed an application to render a kind of content it was not able to normally create itself, and was specifically designed to support compound documents.

Apple, weighed down by its internal power struggle, finally followed with their own object linking model called Publish and Subscribe, released together with the Apple System 7 in 1991. The concepts were inspired by their successful application/development environment HyperCard and allowed content to be linked together across programs, both on a local computer and on a network. Users could publish documents on a shared folder, and other people on the network could subscribe to those documents and incorporate them in their own. Updates to the original data would automatically propagate to all other instances of the file that incorporated it. ClarisWorks, probably the most successful application on the Apple platform, was an integrated package that successfully implemented

[47] Rosch, 'Can Integrated Software Co-Exist with Windows?' p. 60.
[48] Apple Computer (1995). 'Macintosh vs. Windows 95: OpenDoc.' In: URL: http://tech-insider.org/mac/research/acrobat/Mac/950829.pdf.
[49] Hossein Bidgoli (2004). 'The internet encyclopedia (Volume 2).' In: p. 23.
[50] Laurie Flynn (1989). 'Applications for DDE are Starting to Appear.' In: *InfoWorld* 11.44, p. 13.
[51] Ibid., p. 13.

the component philosophy (if not the document-centric approach). Data types were contained in components and could be copied across the different document types, which meant that a text object could be added to a graphics document and the user would be able to access the word-processing capabilities of its parent application through it.

Lotus Development Corp., still limping along after the VisiOn fiasco, also developed an object-linking technology called Link, Embed, Launch-to-edit (LEL). Similar to Microsoft and Apple's technology, it allowed users to copy-paste objects between applications, have changes propagate through the maintained link, and launch the original application by clicking on the embedded object.[52]

Despite considerable investment in these software models by major players in the industry, the systems never quite seemed to materialise. PC Magazine, writing in 1994 about software suites, aptly captured the state these designed seemed to languish in for years: "Lotus, Microsoft, and WordPerfect have all made progress, but the seamlessly integrated suite still seems a version or two away".[53] Exactly why component software failed to become the new dominant design is unclear. Steve Jobs killed Apple's project when he retook control over the company, but its progress had already been sluggish for some time. Lotus Corporation was sold off to IBM in 1995 and never finished building the technology. Microsoft's OLE protocol is still alive and implemented in its Office suite, but third-party applications do not seem to take any advantage of it. Perhaps the protocols simply took too long to be established and the rest of the software market moved on from the idea. Perhaps, after years of competition, it was too difficult for these companies to collaborate and agree on a standard. Regardless, after fifteen years of the entire software industry working towards software integration, the dominant design that crystallised was a pale reflection of the initial goals. Visual integration, in the sense that multiple applications could be used side-by-side, was finally achieved when graphical application managers were accepted, thanks to Microsoft's persistence and long financial breath. But there was no standardisation of commands, menu structures, or interface designs that would allow users to easily transfer skills or programs between software. If there was file compatibility it was more likely to be a historical leftover than a planned feature, and there was virtually no functional interoperability at all, historic or otherwise. In the end, the walled-garden model of software won out, shedding reprogramming the code, writing macros, installing add-ons, or combining the functionality of multiple applications as the last few options users had to negotiate the design of applications.

5.9 CONCLUSION

The application model of software is a construct that was profoundly shaped by its progressive commodification between the 1950s and 1990s. It has resulted in

[52] Doug Barney (1993). 'Object Linking Readied for Unix Notes Clients, Programs.' In: *InfoWorld* 15.24, p. 18.
[53] Michael J. Miller (1994). 'Are They Suites Yet.' In: *PC Magazine* 13.18, p. 159.

a dominant design that centralises control with the software's developers while constraining its negotiability for other stakeholders.

In the early days of mainframe computers, software was not considered to have any inherent commercial value. Users had to write their own code and were free to share it with others. When computer manufacturers realised that software would become a bottleneck for the financial viability of their hardware industry, they invested company resources to support the creation of user-groups: communities of computer users who cooperatively designed their software, reducing development time by eliminating redundant efforts. During this time, software was a collectively negotiated artefact, and the only barriers to participate in that discussion were side-effects of the complexity of the technology.

As the hardware market matured and the share of computers with similar architectures increased, it became possible to reuse previously written code. This meant that it now had some value in and of itself, leading to emergence of the software contractor and software package. Rather than writing the code for the context in which it would be used, companies were now willing to purchase prebuilt software that was customised with the help of the contractor. While still negotiable, this process of software development (and the costs associated with it) made it more common for people to simply accept its design and adjust their operations, rather than the other way around.

While moderately profitable, developers of software packages wanted to increase the value they could extract from their software, so they leveraged the legal system against computers manufacturers who were still providing free software bundled with their other products, and worked towards normalising the idea that software was a product you had to pay for. This transformation further imbued software code with economic value, incentivising the developers to consolidate control over it by implementing the first intentional constraints to the software's negotiability. They obfuscated the source code – making it impossible for users to copy or adapt its design – and required their customers to sign agreements relinquishing full ownership.

In the mid 1970s, the microcomputer appeared as a new platform for a software industry to develop around. Early microcomputer games and applications were collective artefacts whose code was accessible and distributed through magazines for hobbyists to redesign. Once the microcomputer became affordable and approachable enough that general consumers could purchase one, it transformed software into a mass-market product. User-friendliness became an important way to successfully sell to non-technical users, abstracting away the complex insides of software as much as possible and relinquishing the developer from the responsibility to help the user understand and customise their program.

However, the limitations of the microcomputer hardware and the dominant design of the software throttled the growth of the application market, because people could only realistically use two or three programs at the same time. To resolve this bottleneck on their potential revenues, the entire software industry worked towards achieving "software integration". Applications competing for a spot on the user's device built in data transfer mechanisms, functional interoperability, and support for add-ons, all to make it easier for their software to be

used alongside others. Once on top, however, they were quick to use technical and legal means to limit the control others had over their software and thus market share, trampling user's abilities to renegotiate the application's design by reprogramming the source code, extending it through add-ons, or combining it with other applications. Once the dust had settled, users were left with a walled-garden design of software applications that could be used side-by-side visually, but had no interaction technologically.

With both the mainframe and microcomputer, we see the same process of progressive appropriation of control over the software by its developers, and the shrinking ability of others to (re)negotiate its design. Before commodification, when the users and uses of the computer were niche, the distribution of control over the technology was a side-effect of its complexity. As the commercial potential of the technology increased, and value was imbued in the software itself, profit-seeking companies redesigned the technology, law, and business practices to consolidate their autocratic power. More collective imaginations of software facilitated by openness, interoperability, and compatibility were supported when it broadened their market shares, but were aggressively shut down if it meant potentially losing customers and revenue to competitors. Today, software negotiation has devolved into consumer choice between not-so-different products, rather than actual personalisation of the technology itself. Computing technology that was developed under different political economies give us hints about what our computational media could have looked like – France's dirigisme produced the Minitel, Chile's socialism generated project Cybersyn. If we want to change the application model of software, those counterfactuals might serve as guiding lights.

6.

SURVEYING APPLICATION USE IN DANISH KNOWLEDGE WORK

6.1 INTRODUCTION

The labour market in the European Union is changing: work is increasingly dependent on digital competences, with an estimated 90% of jobs requiring some IT skills;[1] non-routine work is becoming more prevalent, as routine tasks are automated or outsourced;[2] and work is decentralised, requiring workers to be more entrepreneurial and collaborative as they engage in project-based contracts with multiple employers.[3]

Instigating and guiding these kind of digital transformations has been a cornerstone of the European Union's economic and social strategy for the past twenty years, and continues to occupy a central position under the von der Leyen commission. Curiously, however, these strategies focus almost entirely on data and skills as the two main components for a digital, globally competitive economy, but ignore the computational tools that workers use on a day-to-day basis to productively leverage those data and skills. Despite the fact that software applications are the mediating artefact, there has been no discussion or investigation into whether the dominant designs and their associated business models are fit for the transformations the EU is striving for. As a result, we know little about what kind of applications are used by the European labour force, how they relate to the digital competences the EU is trying to engender, and their effects on the overall digital working conditions.

This chapter reports on a representative survey of application use by Danish knowledge workers – the most digitalised industry in one of Europe's most digital countries. Thematically, the survey operationalised this topic through the following three sub-questions:

[1] Jacques Bughin et al. (2018). 'Skill shift: Automation and the future of the workforce.' In: *McKinsey Global Institute. McKinsey & Company*.

[2] Gene M Grossman, Esteban Rossi-Hansberg, et al. (2006). 'The rise of offshoring: it's not wine for cloth anymore.' In: *The new economic geography: effects and policy implications*, pp. 59–102; Nir Jaimovich and Henry E Siu (2012). 'Job Polarization and Jobless Recoveries.' In: *National Bureau of Economic Research*. DOI: 10.3386/w18334. URL: http://www.nber.org/papers/w18334.

[3] Irene Mandl et al. (2015a). *New forms of employment*. Vol. 2. Publications Office of the European Union Eurofond, Luxembourg.

- What hardware and software do knowledge workers use to accomplish their main job activities?
- What strategies do knowledge workers use to personalise their software?
- What level of digital competences do knowledge workers have?

Using the answers to these questions, we paint a portrait of the digital characteristics of knowledge workers in Denmark, allowing us to better ground future discussions about the impacts of digitalisation and the direction of digital policy.

6.2 METHOD

6.2.1 *Participants*

A total of 3945 participants answered the survey, segmented to reflect the gender, age, and location distribution of the Danish population (table 4). The intended demographic of this survey was knowledge workers, which we identified as anyone in the three top levels of the Danish version of the International Standard Classification of Occupations (DISCO-08)[4]: managers, professionals, and associate professionals[5].

After filtering for this requirement and incompletes, the sample was reduced to 1608 participants.

		Unweighted		**Weighted**	
		3945	*100%*	*3945*	*100%*
Gender	*Female*	2148	54,4	1966	49,8
	Male	1797	45,6	1979	50,2
Age	*18-34*	866	22	1184	30
	35-54	1637	31,5	1475	37,4
	55-74	1442	36,6	1286	32,6
Region	*Capital city*	1260	31,9	1260	31,9
	Sjælland	577	14,6	567	14,4
	Syddanmark	813	20,6	825	20,9
	Midtjylland	863	21,9	891	22,6
	Nordjylland	432	11	403	10,2

Table 1. Unweighted and weighted participant demographics

[4] White papers and studies variously identify knowledge workers based on the nature of the work activities, the sector or industry of the worker, the level of education required, or occupation. For an overview, see (Brinkley, Rebecca Fauth, and Theodoropoulou, *Knowledge workers and knowledge work: A knowledge economy programme report*).

[5] "Ledelsesarbejde"; "Arbejde, der forudsætter viden på højeste niveau inden for pågældende område" i.e., "Work which requires the highest level of knowledge for the field concerned"; and "Arbejde, der forudsætter viden på mellemniveau" or "Work that requires intermediate level knowledge".

6.2.2 *Materials*

The survey consisted of twenty-four questions, focusing on three thematic subjects: hardware and software tools used, software customisation strategies, and digital competences. The questions about digital competences were taken from the self-assessment version of the European Union's Digital Competence Framework, where a participant can rank their competence level (basic–intermediate–advanced) for different skill categories (information processing, content creation, problem-solving) (see Carretero, Vuorikari, and Punie[6] for the full scale).

6.2.3 *Procedure*

We contracted YouGov to disseminate the questionnaire, a commercial internet survey and data analytics company which maintains a panel of respondents that can be segmented in different ways based on the needs of their clients. Because occupations were not one of the categories that YouGov had pre-recorded about their panel, the first question of the survey asked respondents to self-categorise based on the second-level occupation category of the DISCO-08 framework (e.g., health professional, commercial manager, information and communications technician). Those who fell within the right occupation categories progressed to the rest of the survey.

Because YouGov made a mistake in the logic flow of the survey, a second round of re-contact was necessary to get additional responses for a sub-group of the sample (n = 716 (of 787)).

6.2.4 *Analysis*

Although post-stratification weights were applied to correct for imbalanced response probabilities and ensure the sample approached representativeness, these were not used for the analysis. Because we performed a secondary filtering to remove incompletes and non-knowledge workers – and weight scores are contingent on the other respondents – we used the absolute values for the calculations.

Data from open ended questions, such as the names of the software participants used to accomplish their job, were cleaned using fuzzy matching algorithms in OpenRefine[7].

6.3 RESULTS & DISCUSSION

6.3.1 *The Demographics of Danish Knowledge Workers*

The knowledge workers are not distributed evenly across gender, age, and region compared to the overall Danish labour force (table 2). First, our sample of

[6] S Carretero, R Vuorikari, and Y Punie (2017). *DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use. Publications Office of the European Union EUR 28558 EN, DOI: 10.2760/38842*.

[7] http://openrefine.org/

knowledge workers skews more to men (55,2%) than women (44,8%), which, compared to the overall statistics of the labour force in Denmark, is an inversion of ten percentage points. This shift could be explained by the fact that we use the top three occupation categories as a proxy for knowledge work (i.e., managers, professionals, and associate professionals), where discriminatory hiring practices are likely to suppress the proportion of women. Second, the knowledge workers are twice as likely to be between the ages of 35 and 54 (even though this age category was underrepresented in the sample compared to the overall labour force, which might mean the real proportion is even higher). This is perhaps surprising, considering the fact that the knowledge economy is a recent policy direction, something which the European Union has only started orienting towards since the 2000s: we would expect that the younger generation (18–34) occupies the larger share of knowledge workers. Third, the respondents are concentrated in the capital and central jutland region, which could be explained by the fact that these are the two most populated areas (Copenhagen and Aarhus, respectively) and are home to the most and largest research universities of the country: environments which should increase the relative proportion of knowledge workers over other kinds of jobs.

		Filtered (unweighted)	
		1020	*100%*
Gender	*Female*	457	44,8
	Male	563	55,2
Age	*18-34*	256	25,1
	35-54	537	52,6
	55-74	227	22,3
Region	*Capital region*	382	37,5
	Sjælland	133	13,0
	Syddanmark	171	16,8
	Midtjylland	236	23,1
	Nordjylland	98	9,6

Table 2. Filtered participants demographics

6.3.2 *Education*

The knowledge workers are overwhelmingly highly educated, with nearly 70% of them having a bachelor's or master's degree (fig. 14). Only eighteen respondents have a PhD degree, which is surprising considering (academic) research is a quintessential knowledge intensive occupation and would be expected to have a prominent positions in the sample. Considering the real proportion of workers with PhD degrees is 15%, we assume this is due to non-responses.

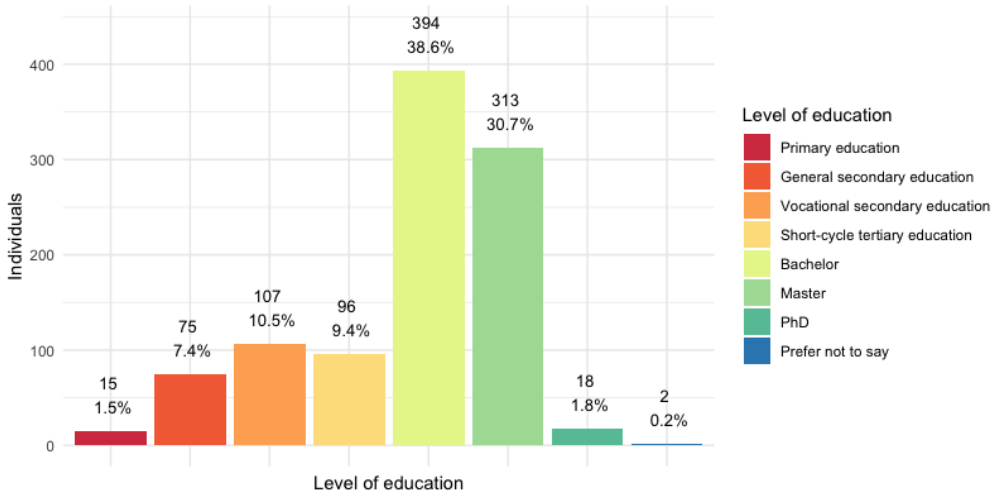


Figure 14. Level of education of Danish knowledge workers)

6.3.3 *Occupation and industry*

In terms of occupation, the majority (57,8%) are "Professionals"[8]; 22,5% are "Associate professionals"[9], and 19,7% are "Managers"). Knowledge workers are

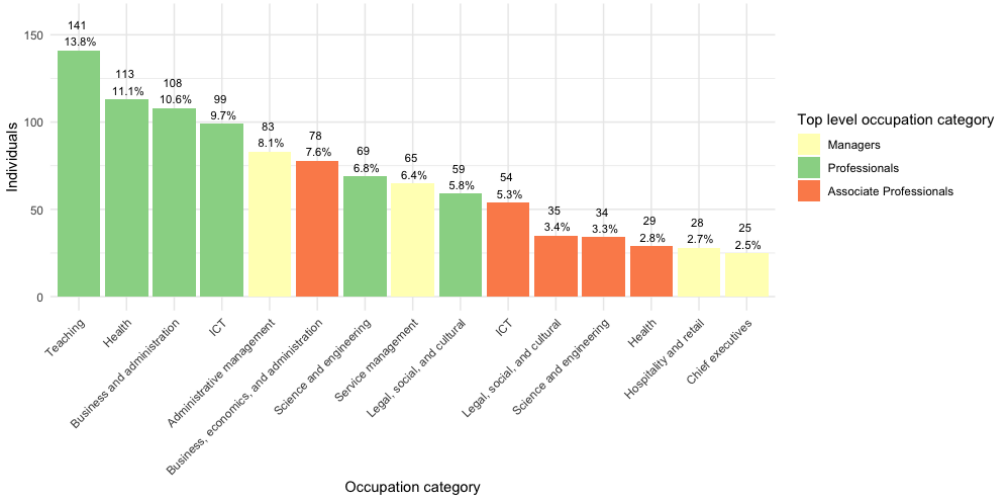


Figure 15. Occupation categories of Danish knowledge workers

most prevalent in occupations that focus on providing services – education, health, public administration, and ICT represent 45,2% of respondents – but are mostly absent in manufacturing industries (oil refinery, mining, furniture). This contradicts the common industry-based distinction that is being made in post-industrial economies between knowledge and service employment; there appears to be a significant overlap, rather than the polarisation that is often presented. Interestingly, the third most popular industry for the respondents was "Other", which

[8] i.e., "Work which requires the highest level of knowledge for the field concerned".
[9] i.e., "Work that requires intermediate level knowledge".

might reflect the prevalence of non-standard employment for knowledge workers, i.e., project-based contracts that move across traditional industry boundaries.

6.3.4 *Hardware*

Nearly 85% of knowledge workers use a laptop or a smartphone for their professional activities (fig. 16). Desktops are less popular, only 60% of respondents use those; and tablets are less popular still, used by just 40% of workers. Overwhelmingly, knowledge workers use just a single device per category, although there is a consistent share of around 17% of workers who use between two or more of the same type of device. In general, the combination of a single laptop and smartphone appears to be the most common tool-set of a Danish knowledge worker.

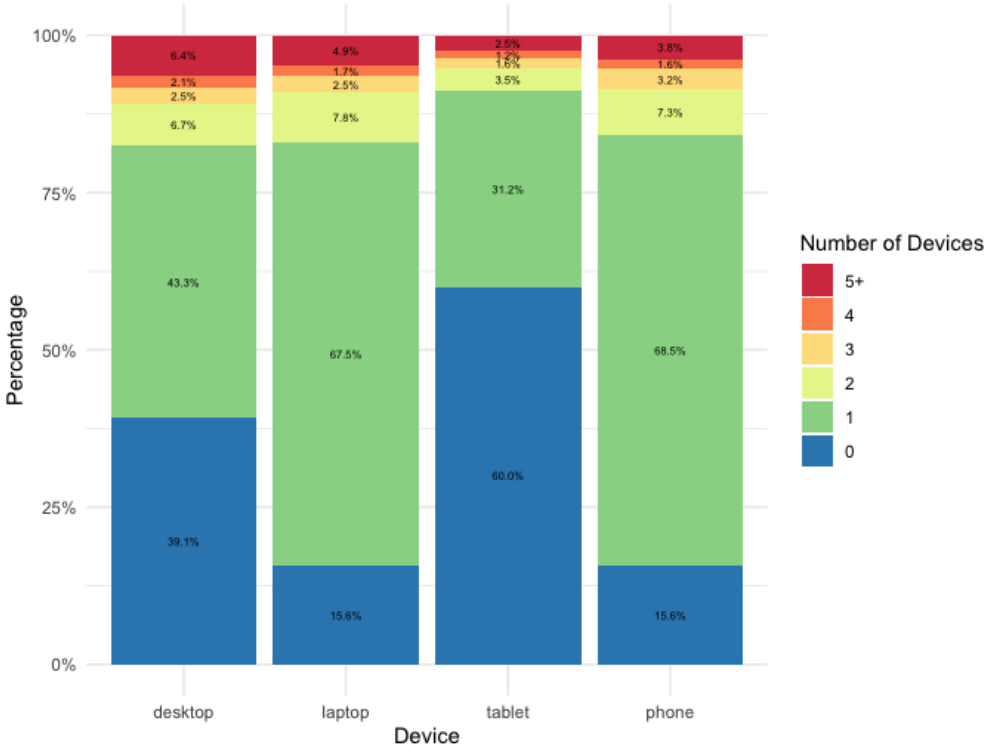


Figure 16. Number of devices used by Danish knowledge workers

6.3.5 *Software*

When asked to list their most important software, 807 respondents provided a total of 2762 (non-unique) applications (fig. 17). This means that, on average, each knowledge worker uses 3,4 different software applications on their laptop/desktop for their main job activities. A little over a quarter of knowledge workers report only using a single application for their job, and 60% use between one and three. In a single, rare case, a worker reported using twenty different pieces of software.

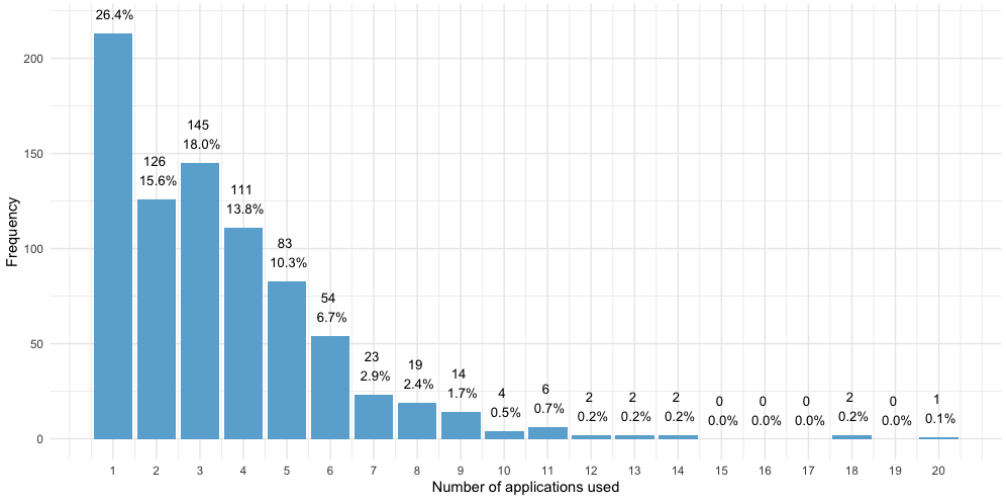


Figure 17. Number of applications used by Danish knowledge workers on their desktop/laptop for their main job activities

There is an extreme homogeneity in the applications used by knowledge workers: the 2762 answers included merely 731 unique software, which translates to an average of 0,91 unique applications per respondent in the 3,4 mentioned. The general pattern is that most workers use the same applications, with the addition of perhaps a single unique one, best visualised as an extreme steep curve with a long tail (fig. 18). The top two mentioned software – MS Word and MS Excel –

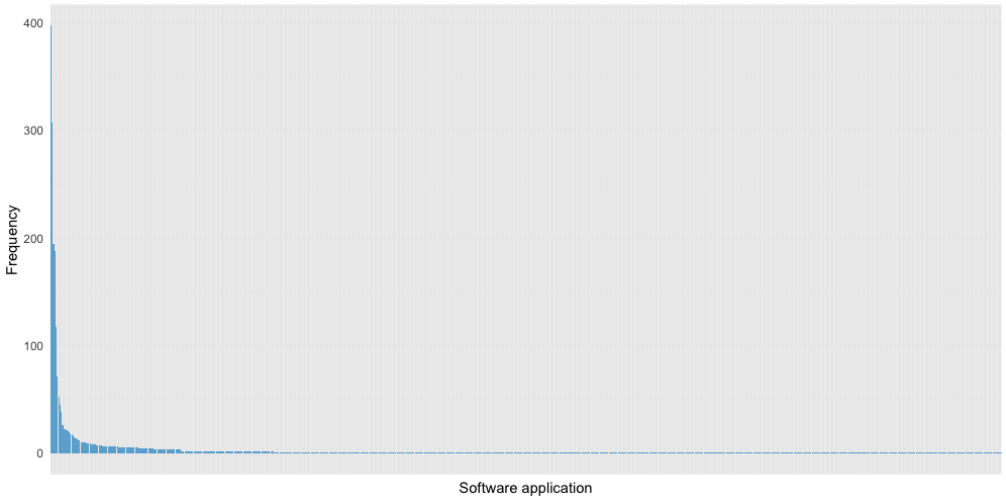


Figure 18. Applications used by Danish knowledge workers on their desktop/laptop for their main job activities

are used by a quarter (25,49%) of knowledge workers (see table 6). The top ten applications are used by more than 50% of all workers.

The lack of diversity is not in the choice of software, but also their characteristics. Of the top thirty applications (representing 62,75% of all software used), twenty-seven are made in the United States, two are from Denmark, and one from Germany. Seventeen – more than half – are designed by Microsoft alone;

four by Adobe, and two by Google. Despite the fact that this software is used to support professional activities, twenty-four of the applications are general purpose consumer applications, and only eight are dedicated business software. Additionally, virtually all applications are produced as a mass-market product. Of the 2762 responses, only 28 reported custom-built software provided to them by their employer or developed by themselves: a mere 1%.

Software	Freq.	%	Cum. %	Origin	Company	Target market
MS Word	397	14,37	14,37	US	Microsoft	Consumer
MS Excel	307	11,12	25,49	US	Microsoft	Consumer
MS Office	194	7,02	32,51	US	Microsoft	Consumer
MS Outlook	188	6,81	39,32	US	Microsoft	Consumer
MS PowerPoint	117	4,24	43,56	US	Microsoft	Consumer
Google Chrome	71	2,57	46,13	US	Google	Consumer
MS Office 365	52	1,88	48,01	US	Microsoft	Consumer
MS Internet Explorer	45	1,63	49,64	US	Microsoft	Consumer
MS Dynamics NAV	38	1,38	51,02	US	Microsoft	Business
SAP	26	0,94	51,96	DE	SAP SE	Business
MS Visual Studio	22	0,8	52,76	US	Microsoft	Consumer
Adobe Acrobat Reader	22	0,8	53,56	US	Adobe	Consumer
MS Skype For Business	21	0,76	54,32	US	Microsoft	Business
Adobe CC	20	0,72	55,04	US	Adobe	Consumer
MS Skype	19	0,69	55,73	US	Microsoft	Consumer
Mozilla Firefox	18	0,65	56,38	US	Mozilla	Consumer
Adobe Photoshop	18	0,65	57,03	US	Adobe	Consumer
MS SharePoint	16	0,58	57,61	US	Microsoft	Business
Citrix	15	0,54	58,15	US	Citrix	Business
MS OneNote	14	0,51	58,66	US	Microsoft	Consumer
Adobe InDesign	13	0,47	59,13	US	Adobe	Consumer
MS Access	12	0,43	59,56	US	Microsoft	Consumer
Autodesk AutoCAD	12	0,43	59,99	US	Autodesk	Consumer
KMD	10	0,36	60,35	DK	KMD	Business
MS OneDrive	10	0,36	60,71	US	Microsoft	Consumer
Sundhedsplatformen	10	0,36	61,07	US	Epic	Business
MS Paint	10	0,36	61,43	US	Microsoft	Consumer
KMD Nexus	9	0,33	61,76	DK	KMD	Business
MS Visio	9	0,33	62,09	US	Microsoft	Consumer
Google Docs	9	0,33	62,42	US	Google	Consumer

Table 3. Top thirty most used application software by Danish knowledge workers

6.3.6 *Software Customisation*

More than half of respondents have, at least once, used the built-in settings (87,6%) and plugins/add-ons (57,7%) to customise their application software

(fig. 30). Almost half (45,8%) have ever used scripts, and a third (31,6%) reprogramming.

Breaking down *whether* knowledge workers customise their software into *how often* they do it, we see that on average they "almost never" do. Of all the different strategies, only the built-in settings are distributed relatively evenly across all levels of frequency; the other strategies are used only sparingly. The proportion of respondents who at least "most of the time" use plugins is a mere 13,2%, dropping down to 6,7% and 4,7% for scripts and reprogramming respectively. It does become apparent that there is a hierarchy between the strategies. As the complexity of the strategy goes up, the proportion of people who "never" use it increases considerably, while all the other frequencies shrink symmetrically. The only category that goes against this pattern is the proportion of people who "sometimes" use plugins, which is higher than those who "sometimes" use the built-in settings.

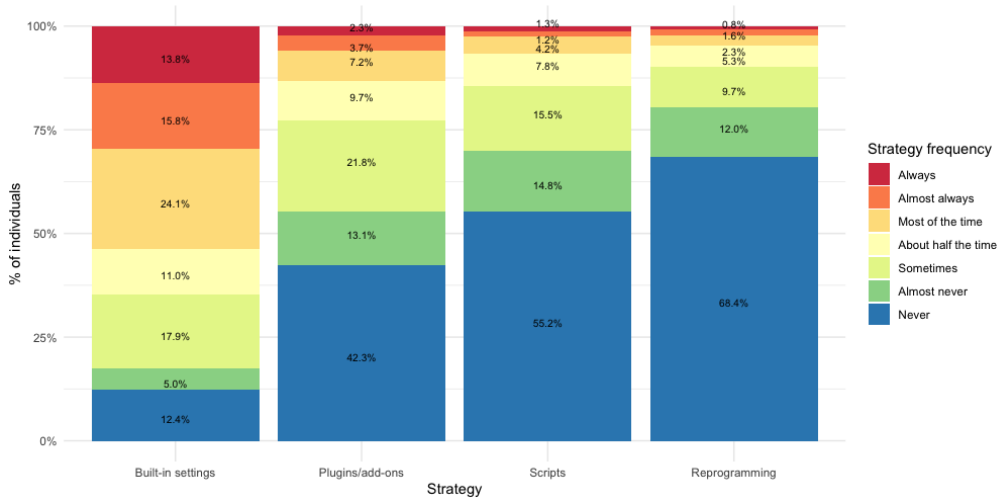


Figure 19. Software customisation frequency across different methods by Danish knowledge workers

Looking at each individual knowledge worker (fig. 20) we can see how the use and frequency levels of customisation strategies are correlated. There appears to be carry-over between the strategies as the complexity goes up: those who use reprogramming "about half the time" are likely the same that use scripts "about half the time". However, this is not uniformly true and there is some transfer between the levels of frequency of each strategy: workers who "never" use plugins, "sometimes" use scripts and those who "never" use scripts "always" use reprogramming.

In summary, the more complex a customisation strategy, the less frequently knowledge workers use it. However, how often a particular worker uses a strategy is not just dependent on how often they use any of the other strategies; there are more variables at play that determine what approach to customisation is used than just its complexity.

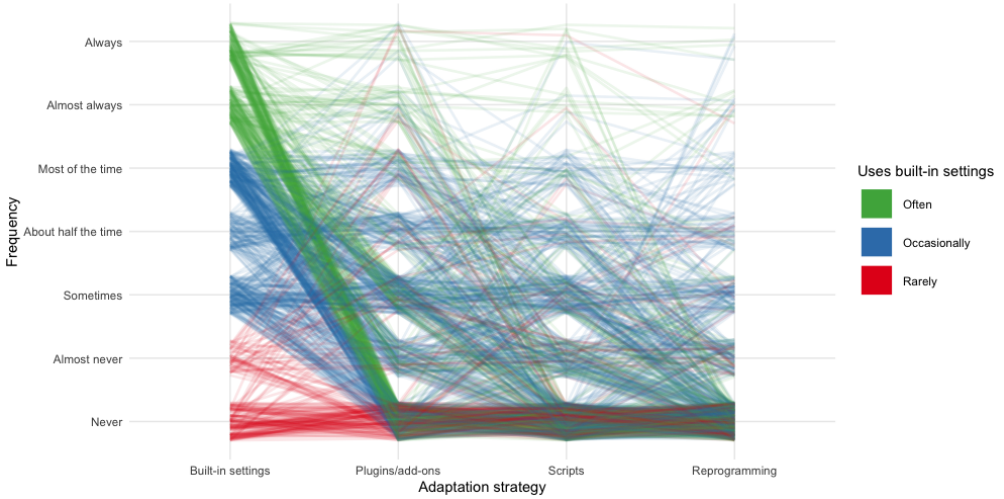


Figure 20. Software customisation frequency across different strategies per Danish knowledge workers

6.3.7 *Digital Competences*

Knowledge workers in Denmark have slightly higher levels of digital competences than the country average. According to the DESI 2020 report, 70% Danish residents have at least basic digital skills, and 49% has above basic skills.[10] Compared to this, 75,8% of knowledge workers have at least basic, and 59,6% have above basic skills (fig. 21).

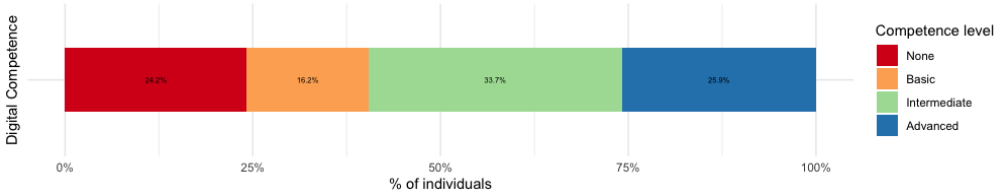


Figure 21. Average digital competences of Danish knowledge workers

6.3.7.1 *Working with digital content*

Digital information is the main material and output of most activities that knowledge workers engage in, which we can see reflected in the digital competences of the respondents. The survey scale used three proxies to measure the ability to work with digital content: *content creation*, *content formatting*, and *computational creation*. Almost 60% of the knowledge workers have at least intermediate level content creation skills and are able to"produce complex digital content in different formats"; a small 14,1% has advanced skills and is able to produce it "using a variety of digital platforms, tools, and environments". The responses to the *content formatting* question are much more polarised. Roughly a third reports having no skills in this category and is unable to "make basic editing to content

[10] European Commission (2020). *Digital Economy and Society Index (DESI) 2020 Denmark*.

produced by others", while another third falls in the advanced category and can "used advanced formatting functions of different tools" such as merging documents of different formats and macros. In terms of *computational content*, it is striking that more respondents report being able to author and control this type of material than format other people's text or images. More than half of the workers "know the basics of one programming language" and 18,9% can "use several programming languages". In other words, digital skills lower in the stack are not fundamental to using more higher level tools: knowing how to program does not automatically confer the ability to mail merge or edit footnotes in an application.

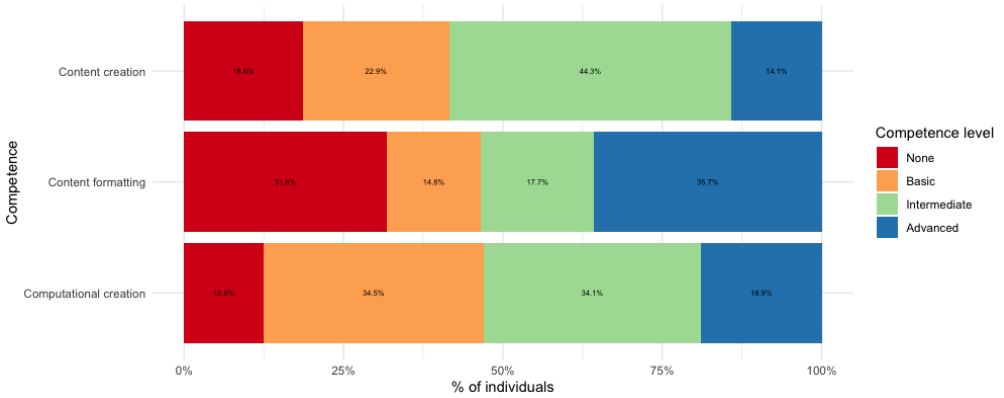


Figure 22. Digital content competences of Danish knowledge workers. "Content creation" refers to generating multimedia data; "Content formatting" to editing other's data; and "Computational creation" to controlling and authoring interactive digital elements (e.g., software settings, code)

6.3.7.2 *Communicating and collaborating with others*

Knowledge work is often done in (distributed) teams with other individuals. Considering this, respondents scored surprisingly low on communication and collaboration competences (fig. 23). 34,5% is unable to "use basic features to communicate" such as sending and receiving emails/text messages, and 41,4% is unable to "share files". Only 15,1% has advanced communication skills and is able to "use a wide range of communication tools" (e.g., instant messaging, social networks); and a quarter advanced collaboration skills such as creating and managing content with online collaborative systems. These results are very counter-intuitive, especially considering the large share of the respondents that use a smartphone, but we are unable to find an explanation for this in the survey design or response quality.

6.3.7.3 *Overcoming and adapting*

According to some definitions, knowledge work can be characterised by non-routine tasks that require continuous innovation and worker autonomy. This also appears to be the most well-developed digital competence of this sample of knowledge workers: compared to the other competences, they score the highest on the questions measuring problem solving and support finding. 67,7% is able to find

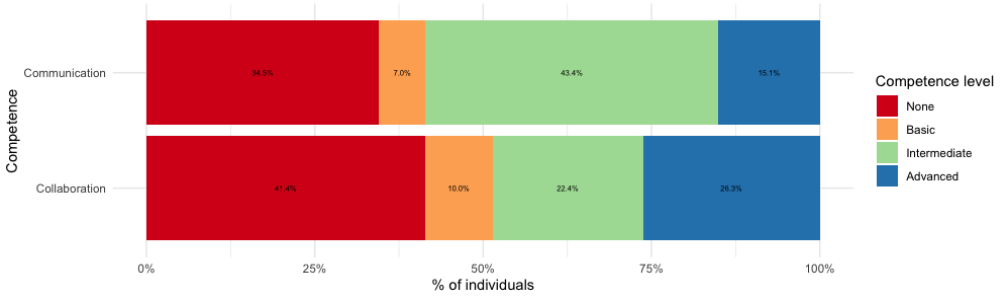


Figure 23. Digital communication and collaboration competences of Danish knowledge workers. "Collaboration" refers to file-sharing and using common information spaces; "Communication" refers to the meta activities to support such activities.

support or by themselves solve "most" or "almost all" problems that arise when using their digital tools. In terms of how they solve their problems, 42,5% has the intermediate-level skills to "explor[e] the settings and options of programs" and 29,8% fall in the advanced category because they "understand the underlying logic of the technology".

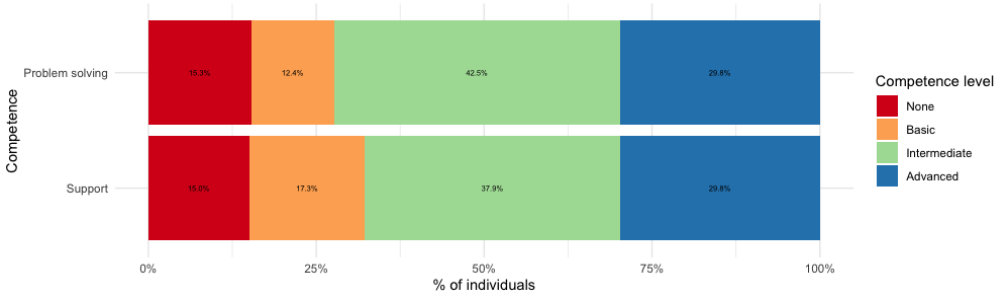


Figure 24. General digital adaptability of Danish knowledge workers. "Problem solving" refers to knowing how to solve unexpected challenges within the context of the tool; "Support" refers to being able to find information and help for problems outside of the tool.

6.4 CONCLUSION

The average Danish knowledge worker is a middle-aged, highly-educated person living in the capital or central jutland region, working in education, health, or public administration. They do their work on a laptop and a smartphone, using three Microsoft Office applications and a single, domain-specific software. They almost never customise their digital tools, but if they do, it is primarily by using the built-in setting options. They know how to make complex documents with multiple types of content using a variety of tools, and might even know a programming language. However, they struggle to use collaborative tools and digitally communicate with co-workers, and find it difficult to edit or format content they did not create themselves. Yet, if they run into technology problems they are

quite capable of figuring out how to resolve them, even if they have not had to deal with them before.

Individual, day to day experiences with the computer inform what Rosenberger calls "relational strategies": the learned ideas about and habits around how to relate to a technology that is stable in a particular way.[11] This survey of application use in Danish knowledge work paints a picture of a digital ecosystem monopolised by a few US American corporations, with a handful of software being responsible for the ideas and habits we develop about computing at large. Rather than the computer as the "intimate supplement" imagined by Bush[12] or the "[hu]man-computer symbiosis" by Licklider,[13] the paradigmatic application model of software seems to be teaching people that a computer contains turn-key products of pre-packaged functionality. With the European Union looking towards the digital economy as the future of the continent, it begs the question whether we want the US to have such outsized control over the artefacts that mediate and cocreate the European labour force.

[11] Robert Rosenberger (2009). 'The sudden experience of the computer.' In: *Ai & Society* 24.2, pp. 173–180.
[12] Vannevar Bush et al. (1945). 'As we may think.' In: *The atlantic monthly* 176.1, pp. 101–108.
[13] J. C. R. Licklider (1960). 'Man-Computer Symbiosis.' In: *IRE Transactions on Human Factors in Electronics* HFE-1, pp. 4–11.

7.

SURVEYING APPLICATION USE IN DANISH KNOWLEDGE WORK

7.1 INTRODUCTION

The labour market in the European Union is changing: work is increasingly dependent on digital competences, with an estimated 90% of jobs requiring some IT skills;[1] non-routine work is becoming more prevalent, as routine tasks are automated or outsourced;[2] and work is decentralised, requiring workers to be more entrepreneurial and collaborative as they engage in project-based contracts with multiple employers.[3]

Instigating and guiding these kind of digital transformations has been a cornerstone of the European Union's economic and social strategy for the past twenty years, and continues to occupy a central position under the von der Leyen commission. Curiously, however, these strategies focus almost entirely on data and skills as the two main components for a digital, globally competitive economy, but ignore the computational tools that workers use on a day-to-day basis to productively leverage those data and skills. Despite the fact that software applications are the mediating artefact, there has been no discussion or investigation into whether the dominant designs and their associated business models are fit for the transformations the EU is striving for. As a result, we know little about what kind of applications are used by the European labour force, how they relate to the digital competences the EU is trying to engender, and their effects on the overall digital working conditions.

This chapter reports on a representative survey of application use by Danish knowledge workers – the most digitalised industry in one of Europe's most digital countries. Thematically, the survey operationalised this topic through the following three sub-questions:

- What hardware and software do knowledge workers use to accomplish their main job activities?
- What strategies do knowledge workers use to personalise their software?
- What level of digital competences do knowledge workers have?

[1] Bughin et al., 'Skill shift: Automation and the future of the workforce.'
[2] Grossman, Rossi-Hansberg, et al., 'The rise of offshoring: it's not wine for cloth anymore'; Jaimovich and Siu, 'Job Polarization and Jobless Recoveries.'
[3] Mandl et al., *New forms of employment*.

Using the answers to these questions, we paint a portrait of the digital characteristics of knowledge workers in Denmark, allowing us to better ground future discussions about the impacts of digitalisation and the direction of digital policy.

7.2 METHOD

7.2.1 *Instrument design*

The survey consisted of a mix of 18 open and closed questions, with a possible maximum of 24 depending on specific conditional answers. The first question of the survey was used to filter respondents based on their occupation, using the sub-major groups of the 2008 version of the Danish International Standard Classification of Occupations (DISCO-08). The rest of the survey was divided into two sections: one with questions about the respondents' use of digital technologies, and one about their demographic characteristics.

The section about digital technologies consisted of questions about the hardware and software they used to accomplish their work activities (which and how many devices, what operating systems, and which software applications for each device); about whether they adapted their software (how often, and using which strategy); and about their digital competences (e.g., digital communication, collaboration, problem solving). The question regarding software adaptation was conceptually informed by partially-overlapping taxonomies developed by,[4],[5] and,[6] resulting in four adaptation strategies: using the software's built-in preference settings, through plugins or add-ons, using scripts or macros, and by reprogramming the source code. The questions regarding digital competences were based on the self-assessment survey of the European Commission's Digital Competence Framework, where a participant can rank their competence level (basic–intermediate–advanced) for different skill categories (e.g., information processing, content creation, problem-solving) (see Carretero, Vuorikari, and Punie[7] for the full scale).

The section about demographic characteristics included questions about employment status (e.g., full-time, self-employed, unemployed, retired), job title, primary work activities, sector (public, private), and industry (e.g., financial and insurance, education, construction). The industry categories were based on the second revision of the Statistical Classification of Economic Activities in the European Community (NACE rev 2).[8] NACE is a multi-level classification with 21 first level categories, each of which is further broken down into more specific activities. This study used 14 of the top level categories, and a selection of the second

[4] Mørch, 'Three levels of end-user tailoring: Customization, integration, and extension.'

[5] Randall H Trigg, Thomas P Moran, and Frank G Halasz (1987b). 'Adaptability and tailorability in NoteCards.' In: *Human–Computer Interaction–INTERACT'87*. Elsevier, pp. 723–728.

[6] Allan MacLean et al. (1990). 'User-tailorable systems: pressing the issues with buttons.' In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 175–182.

[7] Carretero, Vuorikari, and Punie, *DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use. Publications Office of the European Union EUR 28558 EN, DOI: 10.2760/38842*.

[8] EUROSTAT (2008). *NACE rev. 2*. Office for Official Publications of the European Communities. ISBN: 978-92-79-04741-1.

level classifications of 5 other categories. Two categories (sections T and U) were not included.

7.2.2 *Data collection*

7.2.2.1 *Procedure*

The data was collected between July 12 and 22, 2018 by YouGov, a global internet survey and data analytics company which maintains a panel of respondents across multiple demographic characteristics. Respondents earn points for completing surveys which can be exchanged for cash, vouchers, or prize draws.

7.2.2.2 *Participants*

A total of 3944 respondents between the ages of 18 and 74 were contacted, with quotas on gender, age, and region to reach a nationally representative sample.

		Sample		**Population**
		3945	*100%*	*100%*
Gender	*Female*	2148	54,4	49,8
	Male	1797	45,6	50,2
Age	*18-34*	866	22	30
	35-54	1637	31,5	37,4
	55-74	1442	36,6	32,6
Region	*Capital city*	1260	31,9	31,9
	Sjælland	577	14,6	14,4
	Syddanmark	813	20,6	20,9
	Midtjylland	863	21,9	22,6
	Nordjylland	432	11	10,2

Table 4. Unweighted, unfiltered sample and overall population distribution

7.2.2.3 *Variables*

In addition to the data gathered through the survey instrument described in *Instrument design* on 64, the YouGov service included pre-existing background information about the respondents gender, region, age, civil status, and education. The gender data was binary (male, female), and level of education followed the 2015 version of the Danish International Standard Classification of Education (DISCED-15).

7.2.3 *Data processing*

The sample was cleaned to increase the data quality and processed to make it nationally representative.

The data was cleaned based on 1) occupation, 2) non-response, 3) qualitative data quality, and 4) overall response time. 2474 respondents were screened out because their self-reported occupation did not match our definition of knowledge work (i.e., not falling in the DISCO-08 categories of managers, professionals, and technicians and associate professionals[9].). 450 respondents were removed because they did not complete the survey. Respondents were asked to report which software applications they used to accomplish their main job activities per type of device (laptop, desktop, tablet, smartphone). This qualitative data was processed using fuzzy matching algorithms in OpenRefine[10] and manual inspection, resulting in a standardised list of software. All unreasonable answers (e.g., "asdfghjkl", "none") and software names that could not be identified were replaced with the value "-1". All participants with this response for any single, device-related software question were removed from the data set (n = 525). The median response time for the survey was 7 minutes, with the first quartile at 5 minutes and the third at 10 minutes. All respondents with a response time below 2,5 minutes and above 30 minutes were removed (n = 29) After the cleaning, the final sample size corresponds to 466 knowledge workers.
Post-stratification weights were applied to correct for non-responses using the marginal distribution of occupation category separated into sex (female, male) and sector (public, private). Information about the population was retrieved from Danmark Statistik, the official statistics bureau of the Danish government, specifically from "LONS20: Earnings by occupation, sector, salary, salary earners, components and sex"[11]. The weights were calculated using Iterative Proportional Fitting (IPF) or *raking*. Briefly, raking is a method that forces the marginal distribution of a sample to match those of the population by applying a weight to each individual row. It does this by fitting the sample to the population using one demographic statistic at a time (e.g., gender). Once completed, it does the same for the next statistic, until the final distribution equals the population's.

The answers regarding device operating system had to be removed because of a flaw in the conditional logic of the survey that meant respondents were inconsistently shown the question.

7.3 RESULTS

The concept of working environments is rooted in the emergence of the industrial revolution and the dangerous machinery and materials workers were exposed to.

[9] "Ledelsesarbejde" i.e., "Management"; "Arbejde, der forudsætter viden på højeste niveau inden for pågældende område" i.e., "Work which requires the highest level of knowledge for the field concerned"; and "Arbejde, der forudsætter viden på mellemniveau" or "Work that requires intermediate level knowledge".

[10] http://openrefine.org/

[11] Available here: https://www.dst.dk/en/Statistik/dokumentation/documentationofstatistics/structure-of-earnings

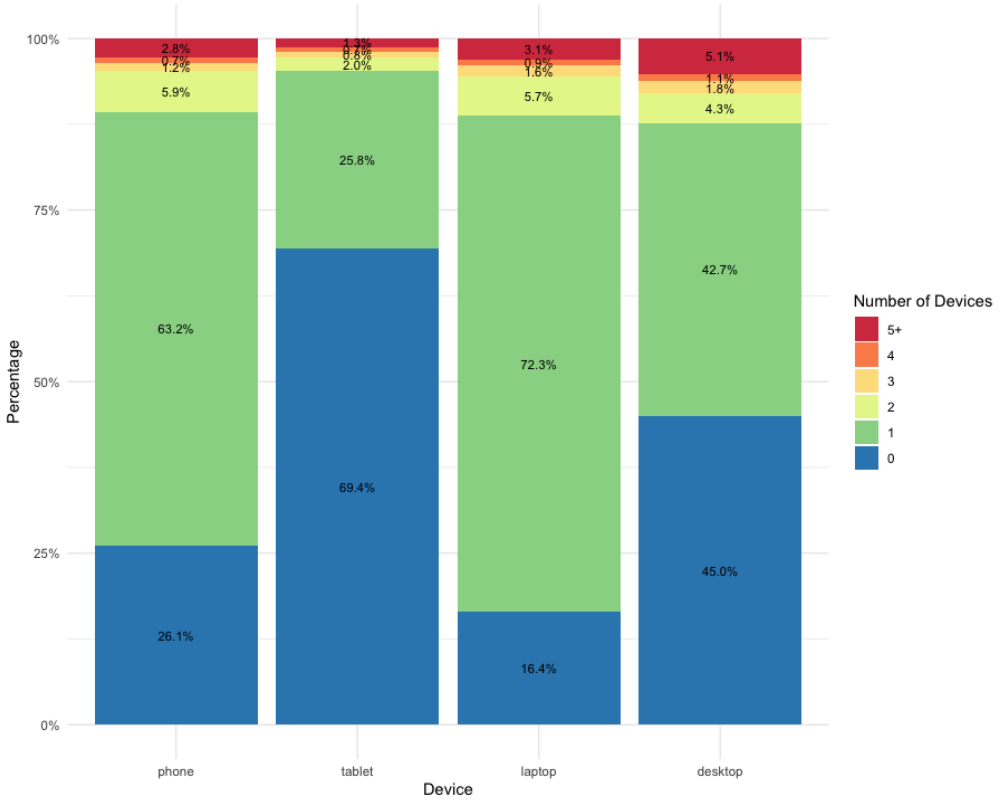


Figure 25. Frequency distribution of different types of devices used by Danish knowledge workers

Since then, it has expanded to not just include physical factors, but also social and cultural aspects that affect the psychological well-being of workers. Digital factors, however, such as the technological devices and the software components, are still missing from most official metrics, even in highly digitalised jobs.

7.3.1 *Hardware Working Environment*

Contemporary knowledge workers have a variety of digital devices to choose from to perform their tasks, ranging from more traditional desktop computers, now-common laptops and smartphones, to the still fledgling tablet form factor.

The survey results indicate that the laptop and smartphone are by far the most common tools for the knowledge worker (see Figure 25). Roughly 83,6% uses laptops, and 73,9% uses smartphones for their professional activity. Desktop computers are less common, bit still used by 55% of workers, and tablets less popular still, used by just 30,6%.

Overwhelmingly, knowledge workers use just one device per category (83,9%), 7,4% report using two copies of the same device type, dropping to 2,2% for three copies and 1,4% for four copies (see Table 5). Interestingly, there appears to be a larger group of workers (5,1%) that use 5 or more copies of the same device.

There are clear correlations in the way these devices are combined (see Figure 26). All devices are combined in some way by a considerable number of workers,

Number of devices	Individuals	Percentage
1	951	83,92
2	84	7,38
3	25	2,22
4	16	1,42
5+	57	5,06

Table 5. Number of devices of the same type (desktop, laptop, phone, tablet) used by Danish knowledge workers

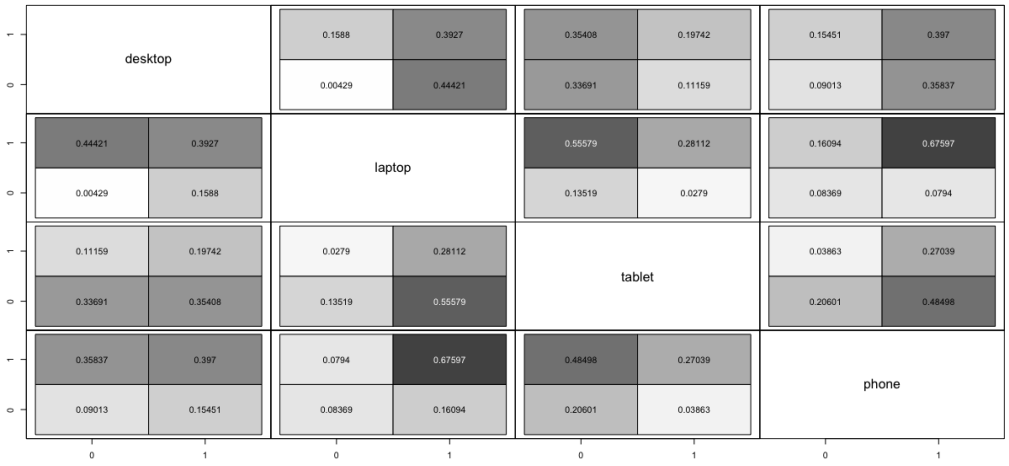


Figure 26. Correlation distribution of different device types by Danish knowledge workers. 0 means the device is not used, 1 means the device is used. The correlations between device and usage can be found by tracing the intersection. The higher the number, the darker the square, the more common the correlation.

with the least frequently used pair being the desktop and the tablet, at just shy of one fifth (19,7%) of the respondents. Pretty much all knowledge workers use either a laptop or desktop for their work – only 0,5% use neither. Almost 40% of workers use *both* a desktop and a laptop, but just as many use a desktop with a smartphone (39,3%). Out of all devices, the laptop-smartphone is the most frequent combination, corresponding to 67,6% of workers, although the laptop is also (and more often than the desktop) combined with a tablet, by more than a fourth of all respondents (28,1%).

7.3.2 *Software Working Environment*

In the 1980s in the United States – the early days of consumer application software – using more than one piece of software as the same time was practically impossible because of hardware limitations such as memory and processing power, but also because of how difficult it was to memorise complicated set of commands

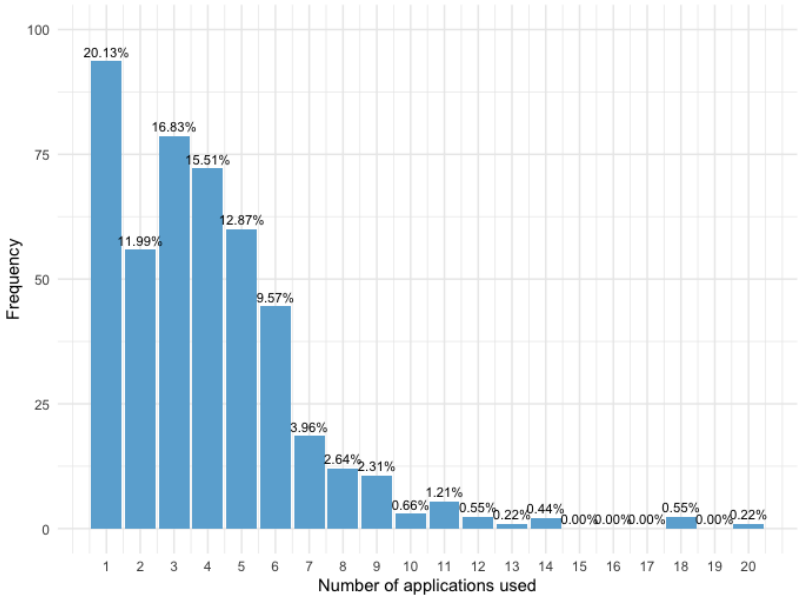


Figure 27. Number of software applications mentioned per respondent as essential to accomplish their work tasks

for more than a handful of applications. These days, in large part because of the invention of graphical interfaces with overlapping windows and continuously improving hardware capabilities, it is technologically possible to use a plethora of applications at the same time. This section reports on the software ecosystems of Danish knowledge workers.

Nearly all respondents (464 out of 466) used either a desktop or a laptop. When asked about the software they use for this device that was necessary to accomplish their work tasks, they mentioned a total of 1832 non-unique applications, with a mean of 3,9 and a median of 4 applications per worker. The largest proportion (20,13%) uses just a single application, nearly half uses between one and three (48,95%), and 86,54% of workers use up to six (see Figure 27).

There is considerable homogeneity in the applications used by knowledge workers: the 1832 answers included merely 535 different software (29%), which translates to an average of 1,15 unique applications per respondent in the 3,9 mentioned. The top two mentioned software – MS Word and MS Excel – are used by a quarter (24,89%) of all knowledge workers, and the top ten applications are used by half (see table 6). The general pattern appears to be that almost all workers use the same (set of) applications, with the addition of perhaps a single unique one: a long-tailed distribution.

The lack of diversity is not just in the choice of software, but also their characteristics. Of the top thirty applications (representing 60,89% of all software used), twenty-nine are made by companies headquartered in the United States and one in Germany. Sixteen – more than half – are designed by Microsoft alone; five by Adobe, and two by Alphabet. Despite the fact that this software is used to support professional activities, many of these applications are general purpose consumer applications, and only seven are marketed as primarily business software (MS Dynamics NAV, MS Skype for Business, MS SharePoint, SAP, MS Access,

Sundhedsplatformen, SAS). Additionally, nearly all applications are produced as a mass-market product. The exceptions are MS Sharepoint, Sundhedsplatformen (the healthcare system for the capital region of Denmark), and SAP, which were either built as custom-solutions or market themselves as being highly configurable to the local environment.

	Software application	Frequency	%	Cum %	Developer	HQ
1	MS Word	256.59	13.92	13.92	Microsoft	US
2	MS Excel	197.19	10.70	24.61	Microsoft	US
3	MS Outlook	131.96	7.16	31.77	Microsoft	US
4	MS Office	91.31	4.95	36.72	Microsoft	US
5	MS PowerPoint	83.12	4.51	41.23	Microsoft	US
6	Google Chrome	52.27	2.83	44.07	Alphabet	US
7	MS Internet Explorer	36.32	1.97	46.04	Microsoft	US
8	MS Office 365	30.46	1.65	47.69	Microsoft	US
9	Adobe Acrobat Reader	19.19	1.04	48.73	Adobe	US
10	MS Visual Studio	18.56	1.01	49.74	Microsoft	US
11	MS Dynamics NAV	18.15	0.98	50.72	Microsoft	US
12	Mozilla Firefox	14.28	0.77	51.49	Mozilla	US
13	MS OneNote	13.81	0.75	52.24	Microsoft	US
14	MS Skype For Business	13.73	0.74	52.99	Microsoft	US
15	Adobe Photoshop	13.14	0.71	53.70	Adobe	US
16	MS SharePoint	13.00	0.70	54.41	Microsoft	US
17	MS Skype	11.73	0.64	55.04	Microsoft	US
18	Adobe CC	10.14	0.55	55.59	Adobe	US
19	SAP	9.89	0.54	56.13	SAP SE	DE
20	MS Paint	9.51	0.52	56.64	Microsoft	US
21	Autodesk AutoCAD	9.35	0.51	57.15	Autodesk	US
22	MS OneDrive	8.53	0.46	57.61	Microsoft	US
23	Google Docs	8.39	0.46	58.07	Alphabet	US
24	MS Access	8.16	0.44	58.51	Microsoft	US
25	Apple Safari	8.03	0.44	58.95	Apple	US
26	Adobe Acrobat Reader XI	7.90	0.43	59.37	Adobe	US
27	Adobe InDesign	7.63	0.41	59.79	Adobe	US
28	Sundhedsplatformen	7.31	0.40	60.19	Epic	US
29	Lotus Notes	6.52	0.35	60.54	IBM	US
30	SAS	6.51	0.35	60.89	SAS Institute	US

Table 6. The top 30 most used applications by Danish knowledge workers

The homogeneity in applications used is also evident in the correlations between applications, as can be seen in Figure 28: there is just a single cluster centred around MS Word, MS Excel, MS PowerPoint, and MS Outlook. There are no independent clusters disconnected from these, which could have represented alternative constellations beyond the Microsoft ecosystem. The internal connec-

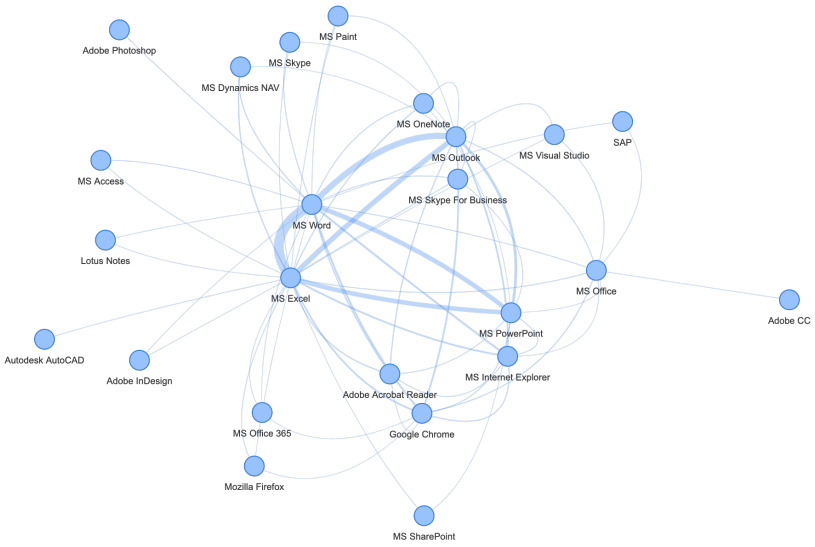


Figure 28. A network visualisation of software applications mentioned together by the same respondent. Only combinations mentioned by at least five workers are included. The thicker the edge connecting two nodes, the more frequently these combinations were mentioned

tions between Microsoft applications seems to show that the software suite is popular for many of its offerings, or that this model helps boost the popularity of one application based on its bundling with the others. Interestingly, this network effect is not present for the Adobe Suite: Adobe Photoshop and InDesign are not connected at all, hinting that these software are used for tasks or occupations with no overlap.

In the outward connections from the core cluster, we can see that these applications are used in combination with software that supplement its functionality (e.g., MS Word with Adobe Photoshop or MS Skype), but also with applications that one could consider alternatives (e.g., MS Word with Lotus Notes or MS OneNote). Similarly, Google Chrome is used in tandem with Mozilla Firefox and Internet Explorer, but the latter are not used with each other. This kind of friendly coexistence does not extend to all software, however. Some applications are clear competitors; MS Skype is never combined with MS Skype for Business, for example, and neither is MS Dynamics NAV with SAP, indicating these are mutually exclusive.

Ultimately, this network visualisation paints the same picture as the overall frequency distribution: the tool set for the Danish knowledge worker is the Microsoft Office Suite, with MS Word and MS Excel the clear power couple.

7.3.3 *Digital Competences*

Digital competences, more reductively referred to as digital skills[12] are seen as one of the core requirements for the successful digitalisation of an industry or

[12] Psychologists conceptualise skills as only one aspect of "the ability to successfully perform a range of tasks to a high level of performance" (Francis Green [2013]. *Skills and skilled work: an*

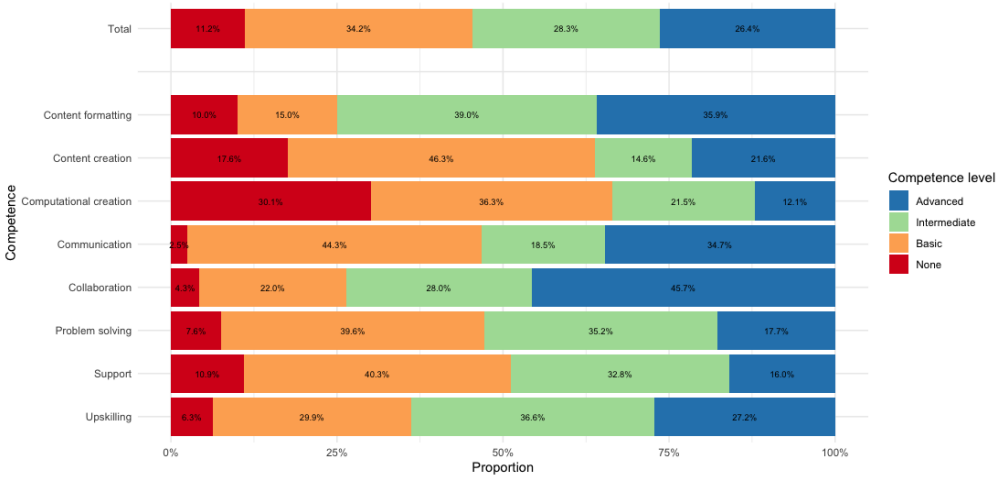


Figure 29. Self-reported digital competences of Danish knowledge workers across eight different types of dimensions

occupation. How exactly to conceptualise and measure these digital competences, however, is still largely unclear. The European Commission has recently proposed a software-independent framework called DigComp, but its fledgling state means there is still little data on the relationship between specific occupations and the presence or requirements of certain competences. This section reports on an early attempt to measure the digital competences of knowledge workers in Denmark.

The respondents of the study have slightly higher levels of digital competences than the country average. According to the Digital Economy and Society Index report of 2020, 58% of Danish residents have at least basic digital skills, and 33% has above basic skills.[13] Compared to this, 34,2% of knowledge workers have at least basic skills, but 54,7% have above basic skills (see Figure 29).

7.3.3.1 *Working with digital content*

Digital information is the main material and output of most activities that knowledge workers engage in, which we can see reflected in the digital competences of the respondents. The survey scale used three proxies to measure the ability to work with digital content: *content creation*, *content formatting*, and *computational creation*.

The respondents are most skilled at *content formatting*: nearly 40% is able to at least "apply basic formatting (e.g., insert footnotes, charts, tables)" to content they or others produced, and 35% can "use advanced formatting functions of different tools" such as merging documents of different formats or applying macros. In terms of creating their own content, just short of half of knowledge workers (46,3%) have basic skills and are able to "produce simple digital content in at least one format", but a sizeable 21,6% has advanced skills and is able to produce "produce or modify complex, multimedia content in different formats using a variety

economic and social analysis. Oxford University Press). The broader concept of competence also includes "knowledge" and "attitude"

[13] European Commission, *Digital Economy and Society Index (DESI) 2020 Denmark*.

of digital platforms, tools, and environments". In terms of *computational content*, this is the dimension where the largest share of workers (30%) report having no competences, in other words, they are not able to "apply and modify simple functions and settings of software and applications". On the other hand, more than a third is able to do this, around 20% knows the basics of one programming language, and 12,1% can use several.

The staggered diminishing of competence levels across these three dimensions meets face-level expectations: editing other people's content is the easiest, followed by creating ones own content. Using more fundamental computer skills such as programming is still far from being the wide-spread competence that most digital policy initiatives are trying to make it. Interesting to note, however, is that despite the obvious importance of creating tangible artefacts that contain the knowledge these workers produced, these three dimensions have the highest overall share of respondents with lower than basic skills.

7.3.3.2 *Communicating and collaborating with others*

Knowledge work is often done in (distributed) teams[14] on a per-project basis, requiring good communication and collaboration skills. This characteristic of knowledge work is reflected in the competence distribution of the respondents. The *communication* and *collaboration* dimensions have the lowest proportion of workers without those skills, 2,5% and 4,3% respectively. Collaboration also has the highest proportion of advanced-level workers, with nearly half (45,7%) able to create and manage content using tools such as electronic calendars, project management systems, and online spreadsheets. In terms of communication competences, 44,3% has the basic skills to use a mobile phone, teleconference, send e-mails, or use chat systems. Roughly a third (34,7%) indicates they "actively use a wide range of communication tools", such as social networks and blogs.

7.3.3.3 *Overcoming and adapting*

By some definitions, knowledge work can be characterised by non-routine tasks that require continuous innovation and creativity.[15] In terms of the use of digital tools, this would require searching for new ways to do things, update ones digital skills in order to explore new ways of working, and being able to handle any technical problems when they arise. The three dimensions associated with these practices – *upskilling*, *problem solving*, and *support* – are the three dimensions that collectively the largest proportion of knowledge workers have intermediate level skills in. Roughly one third is able to "solve most of the more frequent problems" by "exploring the settings and options of programs or tools". Around one third is "aware" that they need to update their digital skills regulary, more than a third is "regularly" doing so, and a bit more than a quarter does this "frequently".

[14] Mandl et al., *New forms of employment*.

[15] Brinkley, Rebecca Fauth, and Theodoropoulou, *Knowledge workers and knowledge work: A knowledge economy programme report*.

7.3.4 *Digital Appropriation*

One of the fundamental tenets of HCI research in general, and practice-oriented CSCW in particular, is that there always exists a gap between the design of a standardised piece of software and the idiosyncratic work practices of the individual/community. This section describes the strategies knowledge workers use to customise their digital tools, and how frequently they use them.

The respondents were asked how often they used the built-in settings, plugins/add-ons, scripts, or reprogramming to adapt their software (see Figure 30). Considering the use of these strategies from a binary perspective, we can observe that 90,87% have used the built-in settings, 59,41% have used plugins/add-ons, 42,47% have used scripts, and 26,64% have used reprogramming.

When going beyond *whether* workers adapt their software and instead consider *how frequently* they do this, the data follows a similar stepwise reduction. A considerable number of respondents (68,42%) use the built-in settings about half the time or more often to adopt their software, but this proportion shrinks to 20,03% for plugins or add-ons, 11,66% for scripts, and a marginal 2,64% for reprogramming. As we move between strategies, which can be considered to grow more complex, the proportion of workers who never use it that strategy increases. In an analogous pattern, as the frequency of using scripts or reprogramming increases, the proportion of respondents is reduced. The use of scripts or add-ons, however, behaves slightly different. Here, more workers "sometimes" use this strategy (24,20%) than "almost never" (14,96%) Of all strategies, only the use of built-in settings is approximately evenly spread across different frequencies (from *Never* to *Always*).

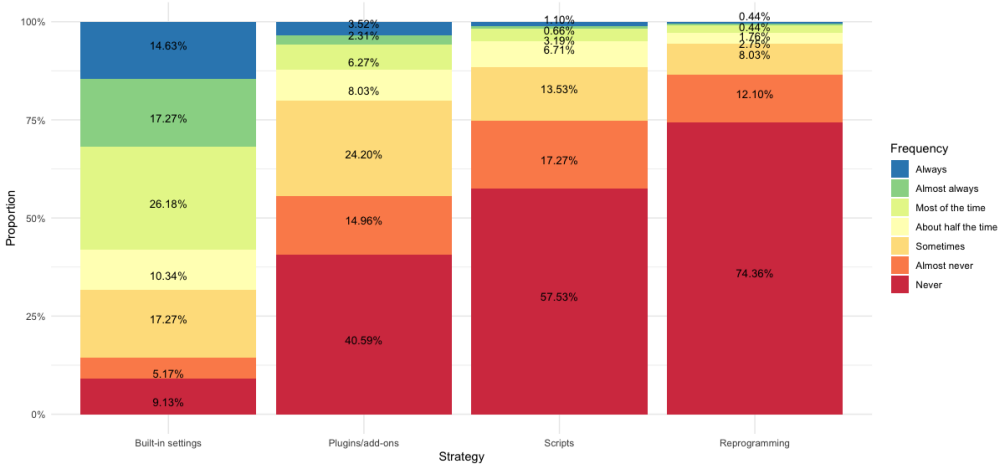


Figure 30. Different software adaptation strategies and how frequently they are used by Danish knowledge workers

The use of certain strategies appear to be correlated with each other in unexpected ways (see Figure 31). Considering the staggered decrease of use going from settings to reprogramming, one would assume that between two strategies, the less complex one is most strongly correlated with the non-use of the other. In other words, if a worker uses the built-in settings, they are more likely to not use

plugins. If they use scripts, they are more likely to not use reprogramming. This does not appear to be the case. Instead, workers that use the built-in settings are most likely to also use plugins, are equally likely to use or not use scripts, and most likely to not use reprogramming. Respondents are roughly just as likely to use plugins and scripting, as they are to use neither; and if they use scripts, they are equally likely to use or not use reprogramming. These correlations suggest that there is some independence between the use of different adaptation strategies: it is not simply a matter of those who use reprogramming also being the ones who use scripting, plugins, and built-in settings. Instead, the data hints at clusters of respondents who combine certain strategies in ways that do not follow their complexity.

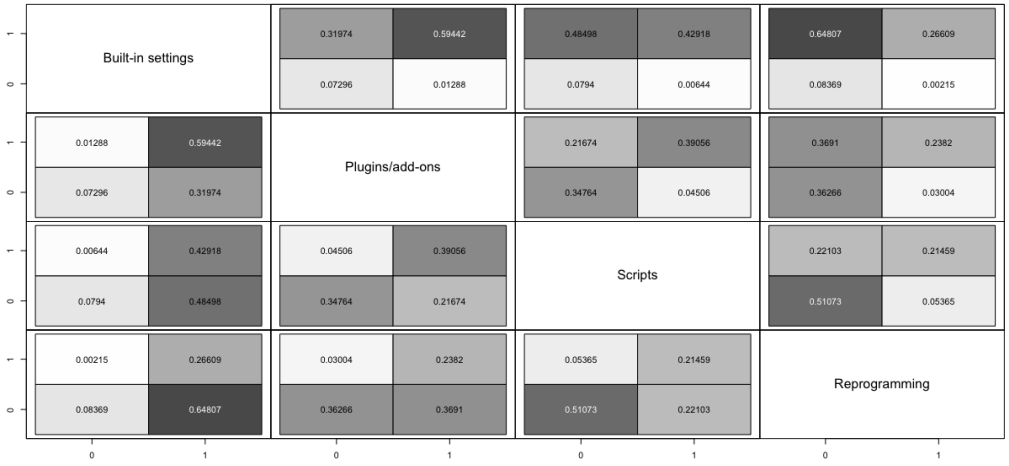


Figure 31. Correlation distribution of different adaptation strategies by Danish knowledge workers. 0 means the strategy is not used, 1 means it is used. The correlations between strategies can be found by tracing their intersection. The higher the number, the darker the square, the more common the correlation.

7.4 DISCUSSION

The average Danish knowledge worker uses a single laptop and smartphone device to accomplish their work tasks. On their main computer, they use approximately four software applications. Like almost all their colleagues, they mostly use MS Word, MS Excel, and MS Outlook, and a single, unique application. When using these applications, they most of the time take advantage of the built-in settings to customise it to their preference, but rarely if ever use plugins, scripts, or reprogramming. Overall, they are comfortable using a computer and know a couple of different ways to approach the same problem using software tools, although there are still areas they are less competent in. They are more skilled at formatting other worker's digital content than creating their own; are comfortable using collaborative tools and know how to communicate with their colleagues using the basic features of a variety of media. If they run into technical problems, they are capable at solving most issues or know how to find support.

7.4.1 *The Dream of Personal Computing*

The computer as an intimate partner, a supplement to the human brain, that might "elevate one's spirit"[16] is a foundational dream of Human-Computer Interaction. Personal accounts of early hobbyists and hackers of the personal computer in the 1970s seem to suggest that such symbiosis were formed, but historiographic analyses of PC magazines from 1980 to 1984 shows how this imagination and relationship transformed as the computer became a mass-market consumer product and the people buying it became *users*:[17] this demographic was more interested in the purposes for which personal computing could be used as a tool, rather than seeing the device as a reprogrammable universal machine.

Our data confirms this tendency and shows that most knowledge workers are *users* of ready-made software that rarely tailor beyond the built-in preferences.

The commodification of software – the emergence of Software as an Application – and the subsequent expansion of its user base with their own diverse visions for the computer (to the chagrin of some computing researchers[18]), requires us to take stock of HCI's dream of personal computing. How close are we to achieving that human-computer interaction? Is it still a worthwhile pursuit, or should it be repositioned as a historic interest rather than one of the main goals of the research community? What design characteristics of contemporary application software is inviting or inhibiting this kind of relationship? What are the wider, structural conditions – the character of the software industry, the increasing geopolitical role of software – that shape the nature of our connection to applications?

As the application software industry emerged, it both stimulated and pursued the imaginary of people as users of computers rather than programmers of computers, of software as a product rather than a medium. One of the early barriers limiting the size of the software product market was how difficult it was to use multiple applications at the same time, and most of the 1980s and early 1990s was devoted to exploring different paths towards the holy-grail of software multi-tasking: application families, integrated packages, windowed application managers, component software, etc. Although Moore's law has mostly eliminated hardware limitations and the graphical user interface has reduced the cognitive strain of learning how to use more than a handful of software, the data from this survey shows that users – or, at least, knowledge workers – still only use between one and six applications. Why is that? Is a few applications simply sufficient to accomplish most work tasks? Or are there specific barriers that inhibit the use of more applications, such as the lack of interoperability or entrenched proprietary document formats? Is it still too difficult to learn how to effectively use more applications, despite the GUI? Or are they not individual factors, but limitations that arise in collaboration with others?

[16] Vannevar Bush (1945a). 'As we may think.' In: *The Atlantic Monthly* 176.1, pp. 101–108.

[17] Laine Nooney, Kevin Driscoll, and Kera Allen (2020). 'From Programming to Products: Softalk Magazine and the Rise of the Personal Computer User.' In: *Information & Culture* 55.2, pp. 105–129.

[18] Alan Kay (2007). 'The real computer revolution hasn't happened yet.' In: *Viewpoints Research Institute* 15.

Another question that arises from seeing which applications are used by knowledge workers is why, despite having largely stayed the same since the 1990s, the Microsoft Office Suite still dominates user's application ecosystem. Is this simply a matter of "the end of history": has Microsoft perfect the designs of word processing, spreadsheet, and presentation software, and are there no reasons to switch to alternative applications? Or are there other forces at play, such as organisational legacies, high (data and skill-based) personal investments, consumer lock-in, or network effect?

7.4.2 *The Geopolitics of Software*

Individual, day to day experiences with the computer inform what Rosenberger[19] calls "relational strategies": the learned ideas about and habits around how to relate to a technology that is stable in a particular way. This survey of application use in Danish knowledge work paints a picture of a digital ecosystem monopolised by a few US American corporations, with a handful of software being responsible for the ideas and habits we develop about computing at large. Rather than the computer as the "intimate supplement" imagined by Bush[20] the "[hu]man-computer symbiosis" by Licklider[21] or software as a "clay of computing" by Kay and Goldberg,[22] the paradigmatic application model of software seems to be teaching people that a computer contains turn-key products of pre-packaged functionality that *you adapt to*, rather than *adapting it to you*. With the European Union looking towards the digital economy as the future of the continent, it begs the question whether we want the US to have such outsized control over the artefacts that mediate and cocreate the European labour force. The predominance of turn key applications leaves little room for workplace democracy traditions in Denmark to have any control over how software is shaped and used.

7.4.3 *The Need for Digital Working Conditions Research*

Regulations of working environments are historically rooted in the physical context that work is performed, designed to protect against dangerous equipment and materials. Since then, a large share of physical labour has become automated or outsourced to other parts of the world, and knowledge and service work has became more prevalent in post-industrial economies. Working environment regulations have evolved with it, now also taking psychological factors that affect worker's well-being into account. The Danish Working Environment Act, for example, takes the broadly construed position that "individual workplaces should be designed in a way which will prevent employees from being forced to leave the labour market due to attrition and stress".[23]

[19] Rosenberger, 'The sudden experience of the computer.'
[20] Bush, 'As we may think.'
[21] Licklider, 'Man-Computer Symbiosis.'
[22] Alan Kay and Adele Goldberg (1977). 'Personal dynamic media.' In: *Computer* 10.3, pp. 31–41.
[23] Arbejdstilsynet (n.d.). *The working environment legislation*. URL: https://at.dk/en/regulations/working-environment-legislation/ (visited on 02/22/2021).

As more and more work becomes digitally mediated, driven on by the sociotechnical imaginary of the digitalised economy and society as the new cornucopia of continued growth and social progress, our conceptualisation of working environments should shift with it to consider the ways digital technologies intersect with the physical and psychological well-being of workers. One could argue that these two higher-order categories are broad enough to also capture the impacts of digital technologies, but without comparative studies between traditional instruments to measure working environments and those that focus specifically on software design, we cannot say for certain whether, or how much, is accounted for. Tentative first steps have been taken across a variety of disciplinary venues, centred around the concept of *technostress*. Ayyagari, Grover, and Purvis[24] describe how the always-on nature of technology, the constant changing nature of software, and the increased ability for worker surveillance are antecedents for later stress. Fuglseth and Sørebø[25] show that the perceived complexity of the software and constant changes are the biggest contributors to technostress, but that technical support and mechanisms that increase worker's digital literacy can have inhibiting effects. Berg-Beckhoff, Nielsen, and Ladekjær Larsen[26] present conflicting results, showing how digital technologies are correlated with stress in cross-sectional studies (which explores bi-directional relations), but not in intervention studies (which would reveal causal relations). However, they do find an association between digital tools and burnout, mostly present in middle-aged working populations. Tarafdar, Cooper, and Stich[27] add a speculative optimistic note, and argues against the prevailing literature to claim that technostress might lead to positive outcomes as well, such as greater effectiveness and innovation. HCI has a clear contribution to make to issues surrounding digital technologies and workplace environments. Current work exploring these questions is not as attuned to interface design, or software models more broadly. The data provided by this study has taken a first step, by trying to representatively capture the hardware and software conditions, and the digital competences and practices related to those factors of Danish knowledge workers.

A better understanding of which elements of software design are causally related to both positive and negative digital working environments can contributed to two agendas. One the one hand, this knowledge can be used to inform digitalisation policies, regulatory initiatives, and – importantly – the instruments currently used to monitor workplace environments. On the other hand, data on which software design elements create or inhibit negative psycho-social experi-

[24] Ramakrishna Ayyagari, Varun Grover, and Russell Purvis (2011). 'Technostress: Technological antecedents and implications.' In: *MIS quarterly*, pp. 831–858.

[25] Anna Mette Fuglseth and Øystein Sørebø (2014). 'The effects of technostress within the context of employee use of ICT.' In: *Computers in Human Behavior* 40, 161–170. ISSN: 0747-5632. DOI: 10.1016/j.chb.2014.07.040.

[26] Gabriele Berg-Beckhoff, Grace Nielsen, and Eva Ladekjær Larsen (2017). 'Use of information communication technology and stress, burnout, and mental health in older, middle-aged, and younger workers–results from a systematic review.' In: *International journal of occupational and environmental health* 23.2, pp. 160–171.

[27] Monideepa Tarafdar, Cary L. Cooper, and Jean-François Stich (2019). 'The technostress trifecta - techno eustress, techno distress and design: Theoretical directions and an agenda for research.' In: *Information Systems Journal* 29.1, 6–42. ISSN: 1365-2575. DOI: 10.1111/isj.12169.

ences can be used to inform the (re)design of commonly used applications. For both agendas, the data from this study can be used to decide which stakeholders to prioritise. Considering the dominance of US American-developed software, and specifically the monopolising position of Microsoft, any regulatory or design interventions should be targeted towards these actors.

7.5 LIMITATIONS

The results from this study should be considered with the following limitations in mind. First and foremost, the data was collected using the commercial survey service YouGov, so the quality of that data is in large part determined by the quality of the panel of respondents they have recruited. In the process of cleaning the data, more than half of the sample was discarded. Although the design of the survey instrument also plays a role, and a conservative filtering method was used, this is still a considerable proportion of the data corpus, and affects the overall confidence in the results. However, it should be noted that the distributions of the answers to the different questions did not always show a considerable change before and after the cleaning (with the exception of the questions about digital competences).

In addition to the quality of the remaining data, the data cleaning also had consequences for the overall sample size, reduced to merely 466 participants. Although the marginal distributions of the sample were close to those of the population, and iterative proportional fitting further aligned the two, the small sample size means that we should be careful when considering the generalisability of the results.

Lastly, the survey instrument was designed for this study, but not validated to confirm that the questions properly captured the intended variables. However, most of the questions included were taken from pre-existing and widely used surveys

7.6 CONCLUSION AND FUTURE RESEARCH

The field of Computer Supported Cooperative Work specialises in providing thick descriptions of technologically-mediated work practices. This paper contributes a nationally representative social survey about the digital working conditions of knowledge workers in Denmark, to contextualise such qualitative data with statistical insights. We collected data on the hardware and software used by knowledge workers, their digital competences, and the extent to which they adapt their software

The analysis show that the hardware and software used by Danish knowledge workers are largely homogeneous. The results demonstrate that products from a few US-based companies have become the de facto standard for computer-mediated knowledge work, and that adaptation of software beyond changing built-in preferences rarely happens.

Considering that the need for local adaptation of software is a basic premise of CSCW research, we highly encourage future work that can shed more light on

this lack of software customisation: is the software simply good enough, or are the costs of appropriation (in terms of time, training, risk of obsolescence) too high? We hope this study encourages more CSCW researchers to consider large-scale survey methods as a worthwhile tool to address these and other questions that provide a high-level overview of the status quo of computer supported work. While their results might not always be shockingly surprising, they complement our qualitatively informed intuitions with detailed empirical data.

8.

THE APPLICATION AND ITS CONSEQUENCES FOR NON-STANDARD KNOWLEDGE WORKERS

Based on Nouwens and Klokmose.[1]

8.1 INTRODUCTION

Application-centric computing has dominated human-computer interactions for the past forty years. Although application-centric computing has been criticised before (e.g., Norman,[2] Bardram et al.[3]), "the application" as a particular way of packaging our interaction with computation has largely escaped empirical investigation: which software characteristics define an application and how they matter is ambiguous. Is it based on the way functionality is monolithically bundled together? On how it operates on the user's document? On its updating and upgrading model? Do those qualities manifest the same across applications? What impact do variations have? As a result, the boundaries of the application are fuzzy, the effects of experiencing almost all computation through them is unclear, and articulating alternatives is difficult.

To unpack and problematise the application as a specific interactive artefact, we ground our discussion in an empirical study of application-use in contemporary knowledge work. The emergence of the application is closely connected to the labour landscape of the 1980s, when large (US American) corporations sought to take advantage of computers to optimise work in various professional domains. But where the technical tendencies of applications have stayed relatively stable, the work practices these applications were intended to support have con-

[1] Midas Nouwens and Clemens Nylandsted Klokmose (2018). 'The Application and Its Consequences for Non-Standard Knowledge Work.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173973. URL: https://doi.org/10.1145/3173574.3173973.

[2] Donald A. Norman (1998). *The Invisible Computer*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-14065-9.

[3] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard (2006). 'Support for Activity-based Computing in a Personal Computing Operating System.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: ACM, pp. 211–220. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124805. URL: http://doi.acm.org/10.1145/1124772.1124805.

tinued to evolve. The labour landscape in post-industrial economies has shifted from predominately full-time, permanent, open-ended employment (i.e., *standard employment*),[4] to one with increasingly more short-term, flexible, project-based forms of labour (i.e., *non-standard employment*).[5] Knowledge work in particular – jobs that involve the creation and distribution of information through non-routine, creative, and abstract thinking – is increasingly performed under these labour conditions: via temporary or fixed-term contracts, as part-time and on-demand jobs, or through self-employment.[6] These non-standard knowledge workers (e.g., freelance journalists, interim managers, and self-employed creative directors) collaborate with (multiple) dynamic groups on a per-project basis for short periods of time across borders and time-zones, mediated through technology. "The application" features prominently in their daily practice as the main environment of production.

How well does the historically situated way of packaging computational interaction as "applications" match contemporary ways of working? How do its challenges and successes relate to the technical tendencies of applications? In what ways do these tendencies manifest themselves in different applications?

This chapter aims to explore "the application" as a class of interactive artefacts through the lens of non-standard knowledge work. The contribution is twofold:

1. It maps out computer mediated collaboration for non-standard knowledge workers, the challenges they face, and the workarounds they employ to circumvent them.

2. It explores contemporary characteristics of applications and the meaning they have in users' experiences.

8.2 RELATED WORK

8.2.1 *Knowledge Work*

Knowledge work has been a classic area of interest in HCI and CSCW. Field-defining works such as Bush's "As We May Think"[7] and the NLS/Augment system[8] took the information society as a central theme and the knowledge worker as the main user to be supported. Engelbart dedicated much of his career on augmenting the "high performance knowledge worker" through developing a "knowledge workshop" system. He explored his vision of reprogrammable software tools to

[4] German Federal Ministry of Labour and Social Affairs (2015). 'Green Paper Work 4.0.' In: *Strategic Notes*.

[5] European Political Strategy Center (2016). 'The Future of Work: Skills and Resilience for a World of Change.' In: *Strategic Notes* 13.

[6] OECD (2015). *In It Together: Why Less Inequality Benefits All*. Paris: OECD Publishing. DOI: http://dx.doi.org/10.1787/9789264235120-en. URL: /content/book/9789264235120-en.

[7] Vannevar Bush (1945b). 'As we may think.' In: *Atlantic Monthly* 176, pp. 101–108.

[8] Douglas C Engelbart (1962). 'Augmenting human intellect: a conceptual framework.' In: *Summary Report, Stanford Research Institute, on Contract AF 49(638)-1024*.

create universally accessible "personal working files"[9] through the framework of Open Hyperdocument Systems and Dynamic Knowledge Repositories.[10]

Much of the research in the area of knowledge work since has further developed and added complexity to these goals. There is a considerable collection of theoretical and ethnographic work on different domains of knowledge work, including scientific collaboration,[11] architectural work,[12] and office management.[13] Other approaches have focused on particular activities within knowledge work, such as file sharing,[14] collaborative writing,[15] using common data repositories,[16] or web activity.[17]

However, few have considered "the application" as the unit of analysis. The ones that have (e.g.,[18]) are mostly rooted in the standard employment perspective and studied workers in large organisations with access to enterprise level and domain-specific applications.

[9] Douglas C. Engelbart (1988). 'Computer-supported Cooperative Work: A Book of Readings.' In: ed. by Irene Greif. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Chap. Toward High-performance Knowledge Workers (Reprint), pp. 67–78. ISBN: 0-934613-57-5. URL: http://dl.acm.org/citation.cfm?id=49504.49507.

[10] Douglas C. Engelbart (1990). 'Knowledge-domain Interoperability and an Open Hyperdocument System.' In: *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*. CSCW '90. Los Angeles, California, USA: ACM, pp. 143–156. ISBN: 0-89791-402-3. DOI: 10.1145/99332.99351. URL: http://doi.acm.org/10.1145/99332.99351.

[11] Marina Jirotka, Charlotte P. Lee, and Gary M. Olson (Aug. 2013). 'Supporting Scientific Collaboration: Methods, Tools and Concepts.' In: *Comput. Supported Coop. Work* 22.4-6, pp. 667–715. ISSN: 0925-9724. DOI: 10.1007/s10606-012-9184-0. URL: http://dx.doi.org/10.1007/s10606-012-9184-0.

[12] Kjeld Schmidt and Ina Wagner (2004). 'Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning.' In: *Computer Supported Cooperative Work (CSCW)* 13.5, pp. 349–408. ISSN: 1573-7551. DOI: 10.1007/s10606-004-5059-3. URL: https://doi.org/10.1007/s10606-004-5059-3.

[13] Abigail J. Sellen and Richard H.R. Harper (2003). *The Myth of the Paperless Office*. Cambridge, MA, USA: MIT Press. ISBN: 026269283X.

[14] Michael Muller, David R. Millen, and Jonathan Feinberg (2010). 'Patterns of Usage in an Enterprise File-sharing Service: Publicizing, Discovering, and Telling the News.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, pp. 763–766. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753438. URL: http://doi.acm.org/10.1145/1753326.1753438.

[15] Bill Tomlinson et al. (2012). 'Massively Distributed Authorship of Academic Papers.' In: *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '12. Austin, Texas, USA: ACM, pp. 11–20. ISBN: 978-1-4503-1016-1. DOI: 10.1145/2212776.2212779. URL: http://doi.acm.org/10.1145/2212776.2212779.

[16] Charlotte Massey, Thomas Lennig, and Steve Whittaker (2014). 'Cloudy Forecast: An Exploration of the Factors Underlying Shared Repository Use.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, pp. 2461–2470. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557042. URL: http://doi.acm.org/10.1145/2556288.2557042.

[17] Abigail J. Sellen, Rachel Murphy, and Kate L. Shaw (2002). 'How Knowledge Workers Use the Web.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, pp. 227–234. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503418. URL: http://doi.acm.org/10.1145/503376.503418.

[18] Richard Harper and Abigail Sellen (1995). 'Collaborative Tools and the Practicalities of Professional Work at the International Monetary Fund.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., pp. 122–129. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223920. URL: http://dx.doi.org/10.1145/223904.223920.

8.2.2 *Non-Standard Work*

Historically, studies on technology in work practices were situated in corporate contexts. In third wave HCI,[19] research followed computers beyond the organisational walls and into non-traditional environments. Nomadic work in particular has received considerable attention, with investigations into such topics as the effect of spatiality in work activities[20] and the struggle these workers face with maintaining their mobile offices.[21]

In addition to different workplaces, mobile technologies have also allowed different employer-employee relationships to emerge, as can be found in some types of non-standard work. Although it lacks a universally agreed upon definition, non-standard work is generally considered to include three forms of employment: temporary or fixed-term contracts, part-time and on-demand jobs, and self-employment.[22] There are multiple, continuously evolving types of work performed under such employment contracts, such as portfolio work (i.e., working for multiple clients at the same time), platform work (i.e., work offered through digital platforms without a fixed working environment), and crowd work (i.e., big tasks that are broken down and digitally distributed over a large amount of workers).[23] Conform to these developments, HCI literature has engaged with the new types work performed under non-standard contracts – such as the gig economy,[24] the platform economy,[25] on-demand work,[26] and crowd work[27] – but has paid less attention to existing professions (e.g., journalism) under these condi-

[19] Susanne Bødker (2006). 'When Second Wave HCI Meets Third Wave Challenges.' In: *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles*. NordiCHI '06. Oslo, Norway: ACM, pp. 1–8. ISBN: 1-59593-325-5. DOI: 10.1145/1182475.1182476. URL: http://doi.acm.org/10.1145/1182475.1182476.

[20] Brown Brown and Kenton O'Hara (2003). 'Place as a Practical Concern of Mobile Workers.' In: *Environment and Planning A* 35.9, pp. 1565–1587. DOI: 10.1068/a34231.

[21] Norman Makoto Su and Gloria Mark (2008). 'Designing for Nomadic Work.' In: *Proceedings of the 7th ACM Conference on Designing Interactive Systems*. DIS '08. Cape Town, South Africa: ACM, pp. 305–314. ISBN: 978-1-60558-002-9. DOI: 10.1145/1394445.1394478. URL: http://doi.acm.org/10.1145/1394445.1394478.

[22] Manos Matsaganis et al. (2016). *Non-Standard Employment and Access to Social Security Benefits*. Directorate-General for Employment, Social Affairs and Inclusion, European Commission.

[23] Irene Mandl et al. (2015b). *New forms of employment*. Vol. 2. Publications Office of the European Union.

[24] Ali Alkhatib, Michael S. Bernstein, and Margaret Levi (2017). 'Examining Crowd Work and Gig Work Through The Historical Lens of Piecework.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, pp. 4599–4616. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025974. URL: http://doi.acm.org/10.1145/3025453.3025974.

[25] Airi Lampinen and Barry Brown (2017). 'Market Design for HCI: Successes and Failures of Peer-to-Peer Exchange Platforms.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, pp. 4331–4343. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025515. URL: http://doi.acm.org/10.1145/3025453.3025515.

[26] Rannie Teodoro et al. (2014). 'The Motivations and Experiences of the On-demand Mobile Workforce.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: ACM, pp. 236–247. ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531680. URL: http://doi.acm.org/10.1145/2531602.2531680.

[27] Lilly C. Irani and M. Six Silberman (2013). 'Turkopticon: Interrupting Worker Invisibility in Amazon Mechanical Turk.' In: *Proceedings of the SIGCHI Conference on Human Factors in Com-*

tions.

Underlying most of these discussions is a sense of precarity and vulnerability of the workers, and the opportunity for specific system designs to promote fairness, safety, and self-determination. For example, Martin et al.[28] describe the invisibility of much of the work Mechanical Turkers do to operate more efficiently and how there is no monetary compensation for it. Glöss et al.[29] discuss how the design of apps – in their case ride-sharing apps like Uber and Lyft – embed particular labour relations, and how this creates responsibilities for the app's designers. Lampinen et al.[30] introduce a "market design" vocabulary and show how it can be used to analyse and generate systems with specific (e.g., enjoyable, equitable) relationships between the market's stakeholders. These and related themes can be seen to culminate in Ekbia and Nardi's[31] recent call for HCI to engage with the political economy of computing.

However, while these contributions acknowledge how technological designs have an impact on labour practices or market structures, "the application" and its specific characteristics are rarely implicated.

8.2.3 *Applications*

Elements of the application model have been challenged from a number of perspectives.

Norman, in The Invisible Computer,[32] criticises applications for being *"homogeneous, super-duper general purpose software packages"* that *"are far too powerful for the use [we] make of them, yet lack all the necessary components for any given task"*. Instead, he proposes information appliances, devices dedicated to a single activity but which are based on a universal information standard so outputs can easily be shared between them.

The Xerox Star implemented a *document-centric* form of computing in which documents operated as containers for bits of applications, allowing the user to combine multiple types of content (e.g., text, graphics, tables, and formulae) that

puting Systems. CHI '13. Paris, France: ACM, pp. 611–620. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470742. URL: http://doi.acm.org/10.1145/2470654.2470742.

[28] David Martin et al. (2014). 'Being a Turker.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: ACM, pp. 224–235. ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531663. URL: http://doi.acm.org/10.1145/2531602.2531663.

[29] Mareike Glöss, Moira McGregor, and Barry Brown (2016). 'Designing for Labour: Uber and the On-Demand Mobile Workforce.' In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. San Jose, California, USA: ACM, pp. 1632–1643. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858476. URL: http://doi.acm.org/10.1145/2858036.2858476.

[30] Lampinen and Brown, 'Market Design for HCI: Successes and Failures of Peer-to-Peer Exchange Platforms.'

[31] Hamid Ekbia and Bonnie Nardi (2016). 'Social Inequality and HCI: The View from Political Economy.' In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. San Jose, California, USA: ACM, pp. 4997–5002. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858343. URL: http://doi.acm.org/10.1145/2858036.2858343.

[32] Norman, *The Invisible Computer*.

remained (mostly) editable in-place.[33] Microsoft's *Object Linking and Embedding* and Apple's *OpenDoc*[34] technology were industry's attempts towards a document-centric paradigm through compound documents, but never quite reached widespread adoption despite significant involvement.

Jef Raskin, in his book The Humane Interface,[35] critiques how applications trap commands in a particular domain and cannot cross application boundaries. Raskin proposes a radically different take on digital content production through the esoteric Canon Cat computer prototype, which provides a command based general purpose content editing environment that replaces files and directories with a searchable database of user content.

The Haystack project[36] argues that there is no way to predict how a user wants to interact with data, so *"there should be no a priori segregation of users' information by 'type' or application"*.[37] Rather than connecting the information to a particular application with a predetermined set of possible operations, their software platform enables users to compound heterogeneous types of data and easily create aggregate views of it.

Bardram et al.[38] describes how applications make combining resources and tools for complex tasks difficult and explored *activity-centric* computing as an alternative, in which the main focus of computation is *"no longer the file (e.g., a document) or the application (e.g. MS Word) but the activity of the user"*. It introduces an activity abstraction on top of regular desktop applications and allows the user to arbitrarily move between computers throughout a working day (e.g., in a clinical setting), and between different applications dealing with the same data types (e.g., PDF viewers on different devices).

Kaptelinin and Bannon[39] problematise the production model of applications, in which changes to digital tools are instigated by the product designers and developers. They suggest moving away from this *product* based application model and instead allow for *"intrinsic practice transformation"*. This means that tool development happens as a result of the ongoing dialectic between the user's practice and the tool, instead of as a result of externally enforced updates and upgrades.

---

[33] Jeff Johnson et al. (Sept. 1989). 'The Xerox Star: A Retrospective.' In: *Computer* 22.9, pp. 11–26, 28–29. ISSN: 0018-9162. DOI: 10.1109/2.35211. URL: http://dx.doi.org/10.1109/2.35211.

[34] Kurt Piersol (1994). 'Under the Hood: A Close-Up of OpenDoc.' In: *BYTE* 19, pp. 183–188. URL: http://archive.li/zPfWW.

[35] Jef Raskin (2000). *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional.

[36] Eytan Adar, David Karger, and Lynn Andrea Stein (1999). 'Haystack: Per-user Information Environments.' In: *Proceedings of the Eighth International Conference on Information and Knowledge Management*. CIKM '99. Kansas City, Missouri, USA: ACM, pp. 413–422. ISBN: 1-58113-146-1. DOI: 10.1145/319950.323231. URL: http://doi.acm.org/10.1145/319950.323231.

[37] David Karger (2007). 'Haystack: Per-User Information Environments Based on Semistructured Data.' In: *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*. Ed. by Victor Kaptelinin and Mary Czerwinski. Cambridge, MA, USA: MIT Press. Chap. 3, pp. 49–100.

[38] Bardram, Bunde-Pedersen, and Soegaard, 'Support for Activity-based Computing in a Personal Computing Operating System.'

[39] Victor Kaptelinin and Liam J. Bannon (2012). 'Interaction Design Beyond the Product: Creating Technology-Enhanced Activity Spaces.' In: *Human–Computer Interaction* 27.3, pp. 277–309. DOI: 10.1080/07370024.2011.646930. eprint: http://www.tandfonline.com/doi/pdf/10.1080/07370024.2011.646930. URL: http://www.tandfonline.com/doi/abs/10.1080/07370024.2011.646930.

While these works discuss "the application" to some extent, they each focus on different characteristics of the software model and base their critiques on a rational evaluation, rather than taking a holistic and empirically grounded approach.

8.3 METHODOLOGY

We conducted an exploratory interview study with 14 participants, focusing on knowledge workers who collaboratively produce textual, visual, or auditory digital artefacts in non-standard forms of employment (i.e., freelance work, self-employment, and fixed-term contracting).

8.3.1 *Participants*

The participants were recruited through snowball sampling until we reached thematic saturation (i.e., until no new categories were found in the last three interviews). The participants ranged from 26 to 57 years old, ten were female and four were male, and they were currently residing in Denmark, The Netherlands, and Belgium. Their employers and collaborators – as is typical in digital labour – extended beyond these country borders. Their employment status varied: six described themselves as freelance, five as self-employed, two were on temporary contracts, and one had a full-time contract but worked as a freelancer before. They had been active as a non-standard worker for different periods of time, ranging from one to sixteen years (mean = 6,4 years). Their occupations were consultant, journalist, sound designer/composer, user experience designer, art director, copywriter, and project officer at an NGO. The participants' type of non-standard work could be categorised as portfolio work: self-managed and income-generating work that is not dependent on a single organisation and where the worker is responsible for building and maintaining relationships with clients from various industries.[40]

8.3.2 *Data Collection*

The interviews were semi-structured and the questions open-ended. Inspired by Flanagan's critical incident technique,[41] questions aimed to draw out specific instances, problems, or highlights of technology use. Participants were asked to walk through a recent or memorable collaborative project, to explain which applications they worked in, what their collaborators were using, what specific problems arose in the process, how it related to their preferred workflow, what workarounds they used, etc. This technique favours detailed descriptions of defined situations over global statements about general use, grounding the participant in their experience rather than asking them to abstract over longer periods of time.

[40] Mandl et al., *New forms of employment*.
[41] John C Flanagan (1954a). 'The critical incident technique.' In: *Psychological bulletin* 51.4, p. 327.

The interviews were conducted in person or via video/audio conference technology (i.e., *Skype*, *appear.in*, or *WhatsApp*), depending on the location or preference of the participant. Each interview was recorded and lasted between 30 and 110 minutes (mean = 55). The recordings were transcribed orthographically; pauses, corrections, and non-verbal interjections (laughter and exclamations) were included. Emphasis and intonation were represented using italics and punctuation.

8.3.3 *Analysis*

The interviews were analysed using thematic analysis, a qualitative analysis method agnostic to theoretical perspectives.[42] Codes and themes were constructed inductively, driven by and closely linked to the data in a bottom-up approach. The transcripts were coded for both semantic and latent themes.[43] For example, a semantic theme included the type of applications participants used to do their work, and a latent theme how the choice of application affected their employability. Themes were topics that existed across the majority of data sets or which were given particular importance within a few data sets (i.e., topics that some participants felt very strongly about). We aimed to provide a rich thematic description of the entire data set to give an idea of the predominant or important themes (as oppposed to focusing on one theme) - some depth and complexity is necessarily lost. Counts (e.g., the number of participants that used *Dropbox*) are reported for the sake of illustrating internal coherency, but should not be seen as representative or used for inference.

8.4 RESULTS

8.4.1 *The Natures of Non-Standard Knowledge Work*

The work described by participants varied in terms of duration, scale, type of activity, and composition of stakeholders. Activities consisted of collaboratively writing news articles for national newspapers, creating soundtracks and sound effects for mobile games, generating content for global marketing campaigns of large companies, writing grant applications for research into food security in Latin America, developing visual brand identities, etc. Projects lasted anywhere between one week to an entire year. Clients included multinational corporations, small agencies, and single individuals. Sometimes participants were able to establish their own team of (non-standard) collaborators for a project, other times they were hired onto an existing team or worked alone. Participants perceived their position within the group of collaborators in different ways: some considered themselves as experts with valuable knowledge and skills, others saw themselves

[42] Virginia Braun and Victoria Clarke (2006). 'Using thematic analysis in psychology.' In: *Qualitative Research in Psychology* 3.2, pp. 77–101. DOI: 10.1191/1478088706qp063oa. eprint: http://www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa. URL: http://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa.

[43] Richard E. Boyatzis (1998). *Transforming Qualitative Information: Thematic Analysis and Code Development*. SAGE Publications, Inc. ISBN: 0761909613.

as a standard employee hired to execute the project brief. Collaborators were rarely located in the same location, but distributed geographically across country borders and time zones (although some met periodically).

Collaboration as described by the participants consisted of three main activities: communication, (co-)creation, and data sharing/storage, each with their own set of associated applications (e.g., *Slack* for communication, *Powerpoint* for creation, *OneDrive* for data sharing). Co-creation happened both synchronously and asynchronously and ranged from symmetrical – where all parties were involved in the same way and to the same degree – to asymmetrical – where some collaborators would contribute only at particular stages of the production. The primary deliverables produced by the participants were always digital artefacts – whether textual, visual, or auditory – but sometimes resulted in physical products or activities.

The applications mentioned by the participants were *Dropbox* (11), *Google Docs* (10), *Microsoft Word* (8), *Microsoft PowerPoint* (7), *WeTransfer* (6), *Microsoft Excel* (4), *Google Slides* (4), *Keynote* (3), *Adobe InDesign* (3), *OneDrive* (3), and *Adobe Photoshop* (2). All participants used the *Microsoft Office Suite*. All but one participant used the *Google Suite*. All five participants working in the visual design industry used the *Adobe Creative Suite*. Applications that were only mentioned by one participant (17) are excluded from this list for brevity.

8.4.1.1 *Digital Characteristics of Non-Standard Work*

Participants repeatedly emphasised how being a non-standard worker shaped their relationship with applications in particular ways. Standard workers can rely on their employer to provide them with (enterprise level) applications, software training, IT support, and top-down enforced standards that (aim to) streamline collaboration (e.g., file-naming conventions, shared repositories, document templates). In contrast, non-standard workers are expected to purchase their own digital tools:

> *As a freelancer, that's just something that you pay for yourself. If you want to use Photoshop, then you have your own Photoshop. You can't expect the client to pay for that.* (P8)

They are hired based on their pre-existing knowledge and are responsible for training themselves in the use of applications:

> *Often a company will make a user for you in whatever system they're using and expect you to become fully aware of how everything works. And that's something I'm always quite reluctant about because spending a lot of time familiarising myself with a system that I'm not going to use again is a waste of my time and I'm normally not getting paid for that.* (P12)

They need to provide their own technical support (or find a more capable peer), even though they have no knowledge of or interest in this role:

> *I'm not the IT guy! I'm not! It's not my expertise. I don't want to be, but I have to be. You have to be everything, it's part of the job.* (P6)

While these expectations can also hold true for standard workers, they take on a different character in non-standard employment. As one participant put it: "*The tools I have need to be more flexible and my knowledge needs to be more advanced than if I wasn't self-employed*" (P3). Whereas a standard worker can be hired on potential, a non-standard worker is expected to have all the required competences. What is normally considered the responsibility of the employer (training, technical support, access to specific tools) becomes a point of competition between non-standard workers. Consequently, the access and skilled use of applications becomes imperative to the worker's position on the labour market and – given the flexibility/instability of non-standard employment – challenges related to these applications can have considerable consequences.

8.4.2 *The Value in Applications*

Eight participants explicitly mentioned how their labour value as a non-standard worker was intertwined with specific applications, based on the skills they developed with and data they invested in them.

8.4.2.1 *Skill-Based Value*

Participants cultivated optimised workflows and personal appropriations with respect to particular applications and considered this part of their value proposition:

> *InVision [44] wasn't meant as a roadmap tool, but for me it is. So that makes me valuable because I thought of a way of using the application in that way. Using something that is at hand and cheap, right here and now, so you don't spend any unnecessary amounts of money and time.* (P7)

Significant changes to the interface or functionality of an application were experienced as undermining this skill-based value. For example, one participant kept postponing an update of her browser-based wireframing application *UXPin* [45] because she was busy and did not feel she had the time to work in a new interface layout. When the update was finally pushed on her, she felt anxious and frustrated:

> *When they make changes, it's like "woah!". It's like I'm starting all over again. It's huge and it's like "aaahh what am I doing here".* (P8)

At the same time, similarities across applications allowed participants to transfer their skills from one application to another, resulting in a blasé attitude towards using unfamiliar applications:

[44] A web app for creating clickable, interactive prototypes based on static images.
[45] A web app for designing, prototyping, and documenting interfaces.

> *Recently I had a big project where we had to reuse some print files and the base of that was done in InDesign. I don't work in InDesign and I don't know that program. It's fairly complicated. But I know the digital logic of Adobe products, so I wasn't really fussed.* (P6)

This shows how uniformity across applications in a software suite or industry at large can help relax the coupling between an application and the embodied skills or workflows users have developed related to that particular application.

8.4.2.2 *Data-Based Value*

Participants created value in applications through investing them with data (e.g., template files, email addresses, purchased plugins):

> *I have different [Google] documents that I use all the time for new clients. I have different presentation templates. I have this document I call the "work document" for working with clients remotely where I maybe put the agenda and a few thoughts, maybe some inspiration, imagery, animations. And I have this scoping document whenever I need to "win" a client. [...] The project management work I do [using these documents], I consider that part of my toolbox. It's a reason why people hire me; it's part of my value.* (P8)

Some participants would try to only work in sustainable file types or explicitly export their files to more universal formats, in an effort to protect their data-based value:

> *I don't save into weird new photo formats that Photoshop just came up with last week. I just stick to the standard formats of things. Because that way there is a bigger chance that you are able to open them and use them.* (P10)

Although not possible with all types of data, these efforts allowed participants to decouple their data-based value (to some extent) from the application and make it accessible beyond its boundaries.

8.4.2.3 *Software Conservatism*

Participants exhibited software conservatism in that they preferred to use applications they were familiar with and had invested in rather than exploring (potentially) better alternatives. One participant was aware of the benefits of cloud-based applications and had even previously compiled a list of recommended services for colleagues. Despite repeatedly emphasising how he does *"everything in the cloud"*, when sharing stories of his actual practice, he stated that he was *"still quite conservative; I'm still writing in Word in a client"* (P4). When communicating with his collaborators, he would email them a twelve year old document that requested them to give feedback through *Microsoft Word*'s track and trace functionality.

However, this loyalty did not guarantee the preservation of their application-based value. Updating or upgrading an application can result in changes that are just as significant as switching between applications. The way these changes are delivered has evolved over the years conform to new business models, from updates as physical products, to digital downloads, to the subscription-based Software as a Service model. Each of these delivery models comes with a different power relationship between the producer and the consumer of the application, with the control increasingly in the hands of the manufacturer to decide when and what to change.

8.4.3 *Personal Preference vs. Collective Compromise*

In collaboration, the use of the participants' value-laden applications was complicated further. All fourteen participants reported multiple instances where there was a tension between wanting to use their preferred applications while simultaneously needing to adapt to the constraints imposed on them by the collaboration. Participants mentioned a wide variety of constraints that influenced which applications were accessible to them during a project, such as technical constraints (e.g., operating system incompatibility) or structural constraints (e.g., company's security policies).

The vast majority of the time, however, the main constraint cited by participants was the diversity of institutional and individual practices, preferences, and proficiencies with respect to applications. One freelance journalist, for example, received feedback on his articles via the review functionalities in *Microsoft Word*, as colour coded text in the document with explanations in a separate email, in voicemails with the client dictating changes, and as scans of paper copies with handwritten annotations.

> *The situation for me as a freelancer, having a lot of customers, a lot of sources, a lot of contacts, doing different things, I* have *to be flexible. [...] People all have different levels of education and different working habits, so they use what they are used to.* (P4)

However, this kind of interpersonal asymmetry was considered fundamentally incompatible with the way applications are designed:

> *So it's really about finding the program or technology that we can all be united in, but that's difficult because we all have our own preferences and our own ways of doing things.* (P7)

Consequently, almost all participants started a new project by explicitly negotiating which applications to use, highlighting the importance of reaching a collaborative compromise.

8.4.3.1 *Application Negotiation*

When describing the process of negotiation, participants repeatedly articulated how their position in the discussion related to their status as a non-standard

worker. Ten participants reported that their capacity as an expert-for-hire meant they had to assume a subordinate position and adapt to their clients: *"As a consultant, you're always trying to work in the customer's way"* (P4). Participants cited wanting to make their clients *"feel safe"* (P6) and *"have a positive experience collaborating"* (P9); not wanting to *"change or disrupt the entire organisation just because of that small workshop or website"* (P8); and that, *"in the end, it is easier for me to adapt than to make them adapt"* (P12).

Four participants expressed that their position as an external expert actually made them feel entitled – and even compelled – to enforce their own application preferences, rather than adapting to the client or collaborator. Especially if they felt it would encroach on their main value proposition as a worker or risk their professional image:

> *I decided some time ago that what I'm doing is unique, which is why I can sell it. So why would I change? Why would I not be the one in control of what I'm selling them? I see it as a bit of giving up if they can change me too much, because then they're not getting what they bought. For me it's just been sticking to my guns and figuring out what works and then reshape the clients instead of the client reshaping me.* (P10)

> *Normally I'll just bend. Because it's not important to me. It's important that it works. But I won't bend when I have to present something in person, because then it's my persona that's there.* (P11)

In the effort to achieve application symmetry, the stakeholders in a project often ended up using the *lowest common software denominator*: the application that was still available after the various constraints were taken into account (device and OS compatibility, privacy restrictions, price, interpersonal skill levels, etc):

> *The thing with KeyNote or Google Slides or something is that it's so easy to use, but if you really want to do it beautifully, I would never use that program. I would love to do that presentation in InDesign but since only 10% of the people involved in that project know how to use it, we made that compromise of working in Keynote. But they happen all the time, those compromises.* (P13)

In collaboration, then, the personal application preferences of the participants were routinely replaced through collective compromise, leaning towards the most traditional applications due to widespread familiarity.

8.4.3.2 *The Cost of Compromise*

The compromises participants had to make in collaboration did not prevent them from meeting their contractual obligations, but they did exact a cost on their labour process. Eleven participants detailed in multiple stories how adapting affected their ability to do their work and, consequently, their perceived worth as a non-standard worker. For example, one participant was forced to use his client's online email system instead of his preferred application, which meant he was unable to access his application-based data:

> *I am hired for my network, but my network is also in my Outlook; I have all email addresses and phone numbers in there. But I can't connect the email address they gave me with my email client. I am hired for my expertise and network but I can't do what I want to do. And this is something as basic as an email application!* (P3)

Another participant was hired under the condition that she would use the client's preferred application – even after she explained she was more skilled in a different one – and felt this affected her value:

> *I like to do it in the programs that I know so that I can do my best work. That's why it can be a bit tricky when your workflow needs to change and you need to use Keynote instead of Google Slides for instance. It changes my whole workflow and I get slower and I feel like I am not worth the money that they pay me.* (P8)

In one case, two collaborators tried to meet in the middle – rather than one party adapting to the other – which compromised both their abilities to work optimally:

> *And she was very honest, she said "I don't normally use this so I don't really know how it works". So she sent [the PowerPoint slides] to me and the pictures were beautiful, but you couldn't write in it... it was just made of pictures. So I tried to change it as well and I'm normally quite good in those programs but she'd done stuff that I hadn't seen before.* (P11)

The cost of compromise also had ripple effects beyond the context of a particular project. Frequently, participants would abandon their preferred application in *anticipation* of collaboration, rather than at the request of the client or colleague:

> *I think I am doing that more and more based on the experiences I've had where I send people Google Docs and then they pasted the text into Word documents. So now I've started to assume that Word is the best choice in many situations because it doesn't cause any disturbance and people are not surprised by a Word document.* (P10)

This self-censorship also affected which applications participants were willing to invest in (even at the cost of their own emotional state):

> *I make a ton of PowerPoints. Its the stupidest thing I ever taught myself because it's so boring, but there's a big need for them and it pays my rent. But PowerPoint is such a shitty program. Microsoft needs to be dragged outside and shot. I haven't even started working in Keynote that I hear is so much easier, because none of my clients would be able to access them. None of my clients use Keynote and that conversion there is between Keynote and PowerPoint doesn't work. So if I do it in Keynote it means I have to redo it in PowerPoint anyway. It's actually a better program, a better product. But I haven't wanted to spend the time on it because I can only sell PowerPoints.* (P10)

The cost, then, does not express itself in the inability of the participants to produce in collaboration, but rather in their inability to consistently use the applications they are most experienced with and in systematically preventing workers to use better applications that can improve their work practices. At the same time, having to constantly switch between applications per project inhibits the repeated and deliberate practice with those applications to develop into an increased proficiency. As the above examples indicate, unless a team of collaborators happen to use the same applications in the same way, there will always be a compromise that leaves a stakeholder bereft of their preferred tool and make it more difficult for them to produce at the level they are capable of.

8.4.4 *Cross-Application Collaboration*

Seven participants developed a number of practices to facilitate a (precarious) type of cross-application collaboration in an effort to still use their preferred applications and protect their value, while simultaneously accommodate their clients and collaborators. Most commonly, collaborators would break up co-creation activities among their preferred applications for as long as possible and then move the data across application boundaries:

> *I usually let the copywriter decide what kind of format they want to write in because it's their desktop. It's their tool. So we usually use Word, but before I send it to a copywriter I like to do my draft of it in Textedit or just in an email.* (P10)

This type of distribution sometimes resulted in a state of confusing document divergence, where it was not clear which version of the document was the 'real' or 'final' one:

> *And you know, just proof reading is a big challenge! When you have that kind of... what's the final sentence on this page? Is it the one in the InDesign document? Or the one in the Keynote document? Who gets to decide?* (P8)

If the data was of a non-proprietary type (such as plaintext), crossing application boundaries required some additional work but was mostly seamless:

> *I wrote two blogposts and sent those to her as Google Docs and asked for her comments and she returned them to me in Word, having used the comments function. It makes it a bit harder because then I have to take the comments from the Word documents into my Google Docs because I continue working in my Google Docs. It's more work for them and more work for me.* (P9)

However, with task-specific applications that produce more particular digital artefacts, the data needed to be transformed. Translating between applications (or between different versions of the same application) was done by exporting the file to a more universal file format and opening it in the target application, at the cost of flexibility and interactivity:

> *So he made the slides in Illustrator and exported them as PDFs and then we built the whole slideshow in PowerPoint. And that prolonged the process because when you find a mistake in the PowerPoint, he had to go back and correct it in Illustrator, export it again, and then upload it.* (P7)

> *I know that if I save an InDesign as a PDF, I will be able to actually open it in Illustrator and actually access the different elements because you can do that. It's gonna take bloody ages and it's such a mess and there's weird boxes that don't do anything all over the place, but that's how you do it.* (P10)

The exported file functioned as an intermediary that could be read by both applications, but at the expense of flattening it to a state that can no longer be operated on in the same way, sometimes affecting the file's integrity. The cross-application collaboration participants can engage in, then, is one that requires a constant back and forth between stakeholders depreciating their produced digital artefacts to a format that is readable to some extent across applications.

8.4.5 *Preferred Alternatives*

Specific software properties emerged as being particularly conducive to non-standard work practices. Interestingly, the properties that participants described as the reason why they preferred one application over another were rarely the features one would consider part of the application's unique selling point. Instead, applications were compared and contrasted based on underlying application infrastructures (e.g., how it handled files) rather than the activity the application purported to support (e.g., making wireframes).

8.4.5.1 *Zero-install*

Multiple participants repeatedly and forcefully praised the application *appear.in*, a simple WebRTC video communication web app. They preferred it not because they thought it had superior video/audio quality or unique features, but because of its almost non-existent onboarding threshold:

> *The thing that doesn't work with Skype for Business is that it takes so many steps to get to the actual business. Whereas with appear.in, you get a link and we're running. It's easier than picking up the phone. That's a massive quality criteria for any solution when you're trying to get someone into something new. If they have to figure out how to even get in there... it's like inviting somebody to your house and saying "I'm not gonna unlock the door, you're just gonna have to get a lock-pick and try to break in".* (P6)

Bypassing the traditional installation requirement of applications was preferred by the participants because it addressed the cost and benefit disparity that is common in application negotiations: the person recommending the application

is the principal beneficiary (since they are already familiar with the software and have it set up) whereas the people adapting to this new application have to pay the cost of installing and familiarising themselves with it. Zero-install applications greatly reduce the time and effort required of the newcomer and thus made it more likely for the application to be successfully integrated.

8.4.5.2 *Links Instead of Files*

Participants preferred URL-based documents over file-based documents for a variety of reasons. Files were seen as a (limited) representation instead of a one-to-one translation of the participant's document, whereas URLs gave access to the actual document as the participant saw them in the context of the application.

> *You have to remember you can't send PowerPoints to a client because if a client doesn't have that font that you're using it'll look weird. So you have to export it as a PDF to send it but then you can't use animations in PowerPoint. So if you have slides with things flying in, you have to simulate the animations by creating twenty slides.* (P7)

Files required the participants to actively share a particular state of the document, whereas URLs could be passively shared and still be changed afterwards:

> *In the beginning we would share presentations with clients as a PDF. But eventually we found out, you know, why not just send the link? And then we could even make small changes 5 minutes before the deadline. We didn't have the stress of saving out the file.* (P8)

URLs were seen as easier to manage than files because they were always the right version of the document, they were easier to share, and they could be bookmarked instead of being lost in the client's inbox:

> *When you only have one Google document and everyone can go in there, then you don't have that discussion of which file is the right one because it's just super visible. There's only one copy.* (P8)

> *InVision gives the customer a link that they can share internally and I think that's* really *valuable to them. That you don't have to send the PowerPoint but that you can just send the link. Even though [a file] is just adding a few more steps – downloading and opening it – it's a hassle. With a link you can just bookmark it in your browser, you don't have to go into your email and find the PowerPoint every time.*

One participant even went as far as to say that it was *"impossible to do modern work where everybody sits with their documents in their local drives"* (P6).

Interestingly, it was not the qualities one would normally relate to cloud-based services that made these applications preferable (i.e., real-time collaboration), but rather the qualitatively different experience of interacting, managing, and sharing a digital artefact in the shape of a URL. It contrasts the static and non-representative snapshot of the document that is a file, with the dynamic and "real" version of the document that is accessed through a URL.

8.4.5.3 *Desktop vs. Browser*

Participants predominately preferred web applications over native applications, but not (necessarily) because they support real-time collaboration or their documents are stored in the cloud. Instead, they pointed to the fundamentally different relationship of native applications and web applications with the underlying hardware and software. Native applications have a larger amount of dependencies and participants struggled with managing compatibility issues between a file and different versions of the same application, between an application and different devices, between an application and different operating systems, and between an application and different versions of the same operating system. Web applications, however, absolved the participants from having to anticipate or even be aware of the technological realities of their collaborators:

> *I prefer web apps and that is what I recommend people to use because I only need to send the link and I don't have to worry about whether someone has the right version.* (P4)

> *I recently came back to InVision because it is online. I don't have to worry whether people are on a Mac or a Windows.* (P7)

Browsers and web standards function as a *de facto* universal operating system that can guarantee a higher degree of interoperability across software and hardware than native applications, and the applications built on top of it have a substantively different character because of it.

8.5 DISCUSSION

Our findings demonstrate the central role of applications in (collaborative) non-standard knowledge work. Participants invested data and developed skills in specific applications, and their value as "human capital" became intertwined with the effective and efficient use of those applications. However, participants struggled to control and access those value-holding applications – both in isolation and collaboration. Compromising on their preferred application meant participants were unable to use tools or techniques they had already developed, had to spend time learning how to use a new application, refrained from or were unable to perform certain tasks, felt clumsy, slow, and incompetent, etc. Some non-standard knowledge workers knew which applications were most commonly used by their (prospective) clients and peers and, to avoid the cost of compromise, eschewed experimenting with or investing in applications that might be easier to use or more powerful. Others tried to continue using their preferred applications and would cross application boundaries by copying content or exporting their work to more universal formats. However, those practices transformed their product into something with a different type of fidelity and interactivity that was less flexible and more time-consuming to manage.

8.5.1 *Application-Application Relationship*

Underpinning the participants' experiences is the ubiquitous silo model of applications, in which interoperability between applications is the exception rather than the rule. This forces users to be unified in their application choices if they want to collaborate, and there is a fundamental tension between this prescriptive symmetry and the asymmetric nature of non-standard work. The individual differences in skill level, responsibilities, or preferences between collaborators are not reflected anywhere in the applications: the expert and the novice, the creative and the manager, the traditionalist and the adventurous are all funnelled into using the same application with largely the same user interface that has the same functionality to operate on the same data type. Because of this ubiquitous characteristic in applications, the majority of collaborations started with an explicit discussion about which set of technological constraints to adopt. It created a mentality of assumed compromise in non-standard knowledge workers, to the extent that they changed their application use in anticipation of collaboration and chose to use the lowest common software denominator in their target market, even when they worked by themselves.

Variations in the application model exist that support more asymmetry. For example, email clients operate through standardised protocols, which allows for a different relationship between applications; users are not constrained by the software their communication partner is using and can shift between different applications with different functionality at their own discretion. Microsoft's *Object Linking and Embedding* and Apple's *OpenDoc* frameworks allowed for a similar type of application-application relationship in productivity software – but with limited success. The participants' practices to cross application boundaries can be seen as a bottom-up attempt to achieve this kind of interoperability. By following the path of least resistance, participants found the file types that had overlap between applications and operating systems and exported, copied, or recorded their documents in those formats. This allowed different stakeholders to use their preferred applications – with all the associated labour value – while still working towards a common digital artefact. However, these more universal file types that could travel between applications transformed the document into something different: Adobe Illustrator files with layers were flattened into PDFs, interactive InVision wireframes were turned into static .jpeg images, Cubase audio tracks were reduced to .mp3 files.

8.5.2 *Application-Document Relationship*

The participants' way of achieving application interoperability relies on another common application characteristic: the distinction between the application and the document. Most applications treat the document as something separate from, yet still defined by and intrinsically coupled to, the application. The document can be independent as a file, yet at the same time requires an application to be created, rendered, and operated on. Which applications are able to interact with the file depends on the way in which it has been externalised. Saving the docu-

ment in the format preferred by the application will result in a file that, when opened, closely resembles the document as it existed inside the application when it was created. However, because most applications' formats are proprietary, this also means that the file usually cannot be operated on by other applications. Saving the document in more universal formats increases the chances that it can be opened by other applications, but moves the document further away from how it originally existed in the application. This tension between truthful externalisation and general accessibility is a result of the "separate yet connected" application-document relationship that is common in applications.

Alternative application-document relationships exist that show how the integration or separation between the two can vary. For example, web applications can be considered to have a closer connection between the application and the document because they use a pass by reference model for sharing documents, rather than the pass by value that is associated with file-based sharing. In this approach, documents are not externalised, but can instead be linked to directly in the context of the application: the application and the document are experienced together, rather than as two separate entities (although web-based services such as Google Drive and Microsoft 365 still allow users to save the document as an external file).

Another type of application-document relationship can be seen in EmbeddedButtons,[46] which was an architecture that allowed arbitrary elements in the document (e.g., lines or text) to be turned into buttons. Those buttons could then interact with other elements in the document (while retaining the ability to be operated on by the application). The buttons were part of the document and could travel between editors, but could also be linked to the application and effectively extend its functionality. This blurs the distinction between the document and the application. On the one hand, the document and application start to merge because the document (in the form of buttons) is becoming part of the application's user interface. On the other hand, the document and the application start to separate because the document could contain all the functionality it needs to operate on itself, within itself. However, the EmbeddedButtons documents needed an application to be rendered, so there was still a distinction between the two.

A change in the application-document relationship has implications for the application-application relationship as well. If the application and the document are merged, users could share a digital artefact without needing to consider whether the receiver has access to the same (version of the) application that can open it. Because the digital artefact knows how to render itself, there would be no need to externalise the document and there would be no loss of fidelity. However, if the application/document is serialised and self-contained, users would no longer be able to open documents in asymmetric applications. The document would be identical for all stakeholders, and there would be no way to achieve the application asymmetry that is possible in the current application model.

[46] Eric A. Bier (1991). 'EmbeddedButtons: Documents As User Interfaces.' In: *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*. UIST '91. Hilton Head, South Carolina, USA: ACM, pp. 45–53. ISBN: 0-89791-451-1. DOI: 10.1145/120782.120787. URL: http://doi.acm.org/10.1145/120782.120787.

Webstrates[47] is a more recent interactive environment that pushes the application-document relationship to the extreme and does not technically distinguish between the two at all; any part of the document can be operated on by the user and be made to operate on any other part in the document, including the user interface. A document in Webstrates, called a webstrate, is expressed in standard HTML. A webstrate is loaded in a web-browser like any other web-page, but local changes to the document (including embedded code and styling) are made persistent and synchronized to all clients of the same page. Klokmose et al.[48] demonstrate that Webstrates has the potential to support asymmetric collaboration: it allows users to interact with the same document through entirely different (and personalised) editors. This application model would allow each stakeholder in a project to use the application they felt most skilled, efficient, and comfortable with. However, these capabilities have only been demonstrated through proof-of-concept prototypes.

8.5.3 *Implications for Research, Development, and Design*

The results of our study highlight problems that have been at the core of HCI since before it was an official field of research, but which are still challenging knowledge workers fifty years later. While document-centric and activity-centric computing have targeted some of these issues, those models do not address all of the problems identified here: the need to support application asymmetry in collaboration, the user's contentious control over changes made to their applications, and the difficulty of transferring embodied skills between applications due to design diversity would still be problematic in those approaches.

From a technical perspective, the limitations of the current application-centric paradigm could be addressed at two different scales. The small-scale approach would be to create applications that slot into the current paradigm, but which subvert the traditional model. Entrepreneurial designers and developers could create simple zero-install tools (such as *appear.in*) that can support different facets of (collaborative) knowledge work with an appropriate power-simplicity trade-off. This approach is reminiscent of the UNIX philosophy,[49] which advocates that application software should be written as simple tools that can be combined to do complex tasks (as opposed to using complex tools to do simple tasks). However, some tasks are too complex and do not easily lend themselves to be distributed as an open-source UNIX tool: cloud-based collaborative systems require expensive infrastructure that currently need a reliable revenue stream.

The large-scale approach is to follow in the footsteps of a project such as Webstrates, and fundamentally rethink the application model and the application-document distinction. This could result in systems that support asymmetric col-

[47] Clemens N. Klokmose et al. (2015a). 'Webstrates: Shareable Dynamic Media.' In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. Charlotte, NC, USA: ACM, pp. 280–290. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807446. URL: http://doi.acm.org/10.1145/2807442.2807446.

[48] Ibid.

[49] Rob Pike and Brian W Kernighan (1984). 'The UNIX System: Program Design in the UNIX Environment.' In: *AT&T Bell Laboratories Technical Journal* 63.8, pp. 1595–1605.

laboration with highly idiosyncratic tools that are specifically developed for individual practices. However, the elephant in the room is the sheer difficulty of addressing such fossilised structures as the application model. Once a technology is introduced, "choices tend to become strongly fixed in material equipment, economic investment, and social habit [and] the original flexibility vanishes for all practical purposes".[50] Alternatives, then, can only be built and tested in isolated bubbles. Potential future research would include establishing microcosms where a possible future can be explored and studied.[51]

The way applications are tied to the labour value of precarious workers also creates a responsibility for HCI researchers to not disrupt it. Novelty is privileged in HCI research and new interfaces and interaction techniques often stand out by their noticeable deviation from the status quo. Engaging with Ekbia and Nardi's call to consider the political economy of HCI[52] forces us to ask who stands to benefit most from this approach. The participants' accounts show how stabilising applications, fixing bugs without radically changing interaction patterns, and standardisation protocols would support collaboration better than new features. Favouring innovation over preservation favours those who can innovate over those who have to maintain. Questions that emerge from this perspective are whether there are alternative update and upgrade models for applications that would minimise the way it jeopardises the embodied skills of users, and how fundamental the tension between the users' needs for stability and uniformity and the current novelty-based paradigm really is.

One of the challenges that remains is figuring out the scope of the problem. The application is near invisible in its omnipresence, which makes it difficult to identify its consequences without something to contrast it with. While this study focused on non-standard knowledge workers, the challenges of application-centric computing are not necessarily limited to this demographic. Standard workers in larger enterprises also have applications imposed on them and find ways of working around it[53] and project-based, cross-organisational collaborations are not uncommon in multinational networked organisations either. Exploring its impact in different work domains, different cultures of use, and different economies will reveal the everyday societal and economic costs of the problems workers face in this anachronistic application model.

[50] Winner, 'Do artifacts have politics?'

[51] Henrik Korsgaard, Clemens Nylandsted Klokmose, and Susanne Bødker (2016). 'Computational Alternatives in Participatory Design: Putting the T Back in Socio-technical Research.' In: *Proceedings of the 14th Participatory Design Conference: Full Papers - Volume 1*. PDC '16. Aarhus, Denmark: ACM, pp. 71–79. ISBN: 978-1-4503-4046-5. DOI: 10.1145/2940299.2940314. URL: http://doi.acm.org/10.1145/2940299.2940314.

[52] Ekbia and Nardi, 'Social Inequality and HCI: The View from Political Economy.'

[53] Mark J. Handel and Steven Poltrock (2011). 'Working Around Official Applications: Experiences from a Large Engineering Project.' In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. CSCW '11. Hangzhou, China: ACM, pp. 309–312. ISBN: 978-1-4503-0556-3. DOI: 10.1145/1958824.1958870. URL: http://doi.acm.org/10.1145/1958824.1958870.

8.6 CONCLUSION

The application represents a particular way of packaging interaction with computation. We explored how the common characteristics of applications reveal themselves through the problems and opportunities experienced by non-standard knowledge workers. We found how the labour value of these workers is tied up with the applications they use based on the skills and data they invest in them and how this relationship affects the efficient, effective, and enjoyable execution of their tasks. However, the technical tendencies of application-centric computing require non-standard knowledge workers to regularly abandon their preferred choice and instead switch to applications they are unfamiliar with or which change their ability to produce. At the same time, not all applications are identical and the extent and way in which users have to adapt can vary, based on the way the application relates to the document, different versions of the same application, other applications, and the operating system. The way application characteristics vary shows how they exist on a spectrum and how alternative models can create qualitatively different user experiences. Given the central position of the application in our interaction with computation, empirically exploring the application-centric computing paradigm is imperative in order to further reveal what defines an application, how variations matter, and what alternatives are yet to be explored.

9.

NEGOTIABLE SOFTWARE: LITERATE COMPUTING WITH WEBSTRATES

Based on Rädle, Nouwens, Antonsen, Eagan, and Klokmose.[1]

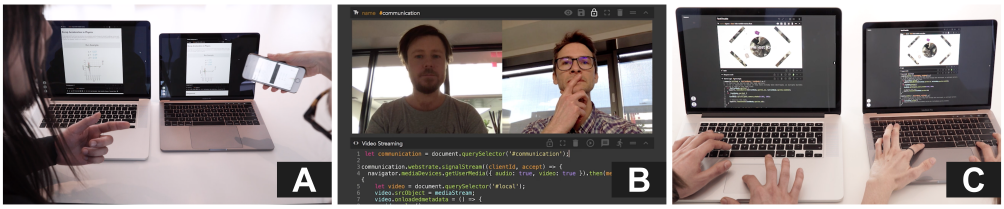


Figure 32. Example uses of *Codestrates*: (A) Collaborative authoring of a physics report. Accelerometer data from a phone is visualized in real-time in the codestrate, and across multiple devices; (B) a codestrate is extended with real-time video communication; (C) the mechanics of a game implemented in a codestrate is collaboratively tinkered with at run-time.

9.1 INTRODUCTION

There is a strong political push in post-industrial economies to consider computational thinking as a "fundamental skill for everyone"[2] is to create a future in which professional and non-professional programmers alike have the skills to develop their own software or adapt someone else's. But computational thinking alone is insufficient: to achieve this goal we need to change how we build software. The current application-centric paradigm of software creates a strong separation between programming and use, making it difficult to renegotiate a program's behavior: most applications offer no meaningful way for end-users to adapt them to their needs. While some allow customization through plugins, they are confined to the boundaries of their specific application's exposed API. Open source applications can be tailored, but require a user to overcome the significant barrier of assuming a programmer role (downloading source code, understanding it, editing, building, and deploying it). If efforts to teach computational think-

[1] Roman Rädle et al. (2017). 'Codestrates: Literate computing with webstrates.' In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 715–725.

[2] Jeannette M. Wing (Mar. 2006). 'Computational Thinking.' In: *Commun. ACM* 49.3, pp. 33–35. ISSN: 0001-0782. DOI: 10.1145/1118178.1118215. URL: http://doi.acm.org/10.1145/1118178.1118215.

ing succeed, we may find that people have the *knowledge* to adapt their digital tools, but not the *power* to do so. We need a new way of building software that is negotiable by design.

Webstrates[3] demonstrates how software can become reprogrammable and extensible in a collaborative fashion, blurring the distinction between applications and documents through a simple change to the web stack—making the Document Object Model (DOM) of web-pages persistent and collaboratively editable. Klokmose et al. present two approaches for developing with Webstrates: 1) using the web browser's built-in developer tools to edit the DOM; and, 2) using a dedicated code-editor webstrate that loads the code of other webstrates through *transclusion* using *iframes*. In both of these cases, there is a separation between using a webstrate and changing its behavior. The first limits development to desktop computers and makes use of a tool meant for debugging rather than developing. The second introduces an application-document relationship between webstrates, where the user has the overhead of loading the target webstrate in a separate code editor to make changes.

In interactive notebooks, use and development happen in the same context. They have become popular with non-professional programmers in education and scientific communities because they allow for authoring content, use code to process data, and visualize results in the same document.[4] Interactive notebooks typically allow for creating documents that interleave blocks of executable code with blocks of (rich) text, and output from the code in the form of textual data or graphics. The creators of the popular Jupyter notebook call this approach *literate computing*[5][6]. Literate computing has the potential for narrowing the gap between developing and using applications, however today's interactive notebooks have limitations: 1) Support for developing applications from within a notebook is limited as it is difficult to save application state, 2) real-time collaboration is, at best, limited to text editing and 3) the behaviour of a notebook cannot be reprogrammed or extended from within, hereby limiting its expressive power.

We present *Codestrates*, an alternative approach to building user-extensible collaborative interactive systems that combines the possibilities of Webstrates with literate computing. Firstly, *Codestrates* pushes the literate computing approach further by making collaborative computation, extension, and development of applications with persisted state possible in the same environment, hereby narrowing the gap between development and use of interactive systems. Secondly, *Codestrates* enables prototyping in a manner similar to code playgrounds (e.g.,

[3] Clemens N Klokmose et al. (2015b). 'Webstrates: shareable dynamic media.' In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, pp. 280–290.

[4] Thomas Kluyver et al. (2016). 'Jupyter Notebooks–a publishing format for reproducible computational workflows.' In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87–90.

[5] K Jarrod Millman and Fernando Pérez (2014). 'Developing open-source scientific practice.' In: *Implementing Reproducible Research* 149.

[6] Knuth's *literate programming* (Donald E. Knuth [May 1984]. 'Literate Programming.' In: *Comput. J.* 27.2, pp. 97–111. ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. URL: http://dx.doi.org/10.1093/comjnl/27.2.97) is the act of interweaving code and documentation in the same source file, where *literate computing* is the blend of writing and executing code with authoring text and rich media in the same interactive document.

Codepen or JSFiddle), but as application state persists, prototypes become usable applications. Thirdly, *Codestrates* provides Webstrates with a development environment that goes beyond the paradigmatic application-document model. *Codestrates* is open source and ready for everyone to tinker at https://codestrates.org.

After reviewing related work, we explain the concept of *Codestrates* and demonstrate it with three use cases, inspired by our own day-to-day use: (i) using a codestrate as a collaborative interactive notebook (fig. 32(a)), (ii) extending the functionality of a codestrate (fig. 32(b)), and (iii) developing reprogrammable applications in a codestrate (fig. 32(c)). We subsequently explain the technical implementation of *Codestrates*, discuss its limitations, and evaluate it based on Olsen's solution viscosity criteria.[7]

9.2 RELATED WORK

Codestrates combines real-time, web-based collaborative authoring, documents that blend multimedia with executable code in a literate computing style, and end-user (re)programmability of the document and the application. We discuss related work that combines some of these elements.

9.2.1 *Collaborative systems and documents*

Jupiter[8] was an early collaborative multi-user dungeon built around shared, persistent "virtual places" (i.e., rooms). Users could create and customize places, documents, and tools in Jupiter using the provided high-level windowing toolkit or the internal programming language. Since then, Google Docs has established itself as one of the first web-based word processors that offered real-time collaboration. Dropbox recently released their own collaborative word processor called Paper[9], which goes beyond traditional documents by allowing users to write text, embed rich media, and include (non-executable) code snippets.

9.2.2 *Scriptable and reprogrammable applications*

9.2.2.1 *Shareable and malleable applications*

HyperCard[10] was an early hypermedia system for producing software that could easily be shared with and adapted by others. Through a visual drag-and-drop interface, end-users could create applications by building "stacks" of interactive cards. Users could programmatically add interactivity to the cards (e.g., a button

[7] Dan R. Olsen Jr. (2007). 'Evaluating User Interface Systems Research.' In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, pp. 251–258. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294256. URL: http://doi.acm.org/10.1145/1294211.1294256.

[8] David A. Nichols et al. (1995). 'High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System.' In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. Pittsburgh, Pennsylvania, USA: ACM, pp. 111–120. ISBN: 0-89791-709-X. DOI: 10.1145/215585.215706. URL: http://doi.acm.org/10.1145/215585.215706.

[9] https://paper.dropbox.com/ (last accessed: July 11, 2017)

[10] D Goodman (1988). *The complete Hyper Card Handbook. Bantam Books*.

on a card could link to another card in the stack) using the provided scripting language Hypertalk. However, HyperCard was only scriptable and not fully reprogrammable.

9.2.2.2 *Reprogrammability of an environment at run-time*

Smalltalk programming systems like Squeak[11] or Pharo[12] allow users to extend a program's functionality or even reprogram it entirely at run-time through just-in-time compilation and late binding.[13] Smalltalk relies on an image-based persistence model which forgoes a hard distinction between system code, application code, and application state. In the recent Lively project,[14] the concepts of Smalltalk have been ported to the modern web architecture using JavaScript. It takes an object-oriented approach to UIs based on Morphic[15] by abstracting over the DOM and CSS.

9.2.2.3 *Web-based dev playgrounds & reactive programming*

Web-based programming environments like JSFiddle[16], JS Bin[17], and Codepen[18] allow the user to experiment with code and rapidly develop UI, functionality, features, or applications that stay reprogrammable and can be shared with others. However, (i) the persistence and sharing of application state is not supported, (ii) the user's code cannot change the development environment (e.g., to increase its expressive match[19] by customizing its tools to match a programmer's personal preference), and (iii) collaboration is only possible for the editing of code.

Various web-based systems exist that try to make application development more approachable by going beyond traditional programming, such as through spreadsheet-like environments in Gneiss[20] or using only HTML in Mavo.[21] How-

[11] Dan Ingalls et al. (Oct. 1997). 'Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself.' In: *SIGPLAN Not.* 32.10, pp. 318–326. ISSN: 0362-1340. DOI: 10.1145/263700.263754. URL: http://doi.acm.org/10.1145/263700.263754.

[12] http://pharo.org/ (last accessed: July 11, 2017)

[13] Adele Goldberg (Oct. 1995). 'Why Smalltalk?' In: *Commun. ACM* 38.10, pp. 105–107. ISSN: 0001-0782. DOI: 10.1145/226239.226260. URL: http://doi.acm.org/10.1145/226239.226260.

[14] Antero Taivalsaari et al. (2008). *Web Browser As an Application Platform: The Lively Kernel Experience*. Mountain View, CA, USA; Robert Krahn et al. (2009). 'Lively Wiki a Development Environment for Creating and Sharing Active Web Content.' In: *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*. WikiSym '09. Orlando, Florida: ACM, 9:1–9:10. ISBN: 978-1-60558-730-1. DOI: 10.1145/1641309.1641324. URL: http://doi.acm.org/10.1145/1641309.1641324.

[15] John H Maloney and Randall B Smith (1995). 'Directness and liveness in the morphic user interface construction environment.' In: *Proceedings of the 8th annual ACM symposium on User interface and software technology*. ACM, pp. 21–28.

[16] https://jsfiddle.net/ (last accessed: July 11, 2017)

[17] https://jsbin.com/ (last accessed: July 11, 2017)

[18] http://codepen.io/ (last accessed: July 11, 2017)

[19] Olsen, 'Evaluating User Interface Systems Research.'

[20] Kerry Shih-Ping Chang and Brad A. Myers (Apr. 2017). 'Gneiss.' In: *J. Vis. Lang. Comput.* 39.C, pp. 41–50. ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2016.07.004. URL: https://doi.org/10.1016/j.jvlc.2016.07.004.

[21] Lea Verou, Amy X. Zhang, and David R. Karger (2016). 'Mavo: Creating Interactive Data-Driven Web Applications by Authoring HTML.' In: *Proceedings of the 29th Annual Symposium on User*

ever, reprogramming finished applications requires external editors or importing it back into the development environment, rather than the use environment.

9.2.3 *Interactive notebooks using literate computing*

9.2.3.1 *Interactive notebooks*

Jupyter notebook (formerly IPython Notebook)[22] and Apache Zeppelin[23] are two popular interactive notebooks that can embed code in multiple programming languages. Interactive notebooks can be used for data cleaning and transformation, numerical simulation, statistical modeling, and machine learning. However, the scope of the code extends only to the content of the notebook: the user interface cannot readily be extended from within a notebook. Jupyter supports extensions, but these have to be installed on the server side and are developed externally to the notebook. Zeppelin supports collaborative editing, but only to designated text areas. Neither Jupyter or Zeppelin are designed for developing state-full applications. Persisting data requires manually writing data to the file-system of the host computer or through a database interface.

9.2.3.2 *Reprogrammable applications using literate computing*

Leisure[24] and Eve[25] are systems that share the most with *Codestrates*. Leisure is an open source, web-based, and document-centric approach to computing based on the Emacs org-mode document format.[26] Leisure provides two-way bindings between an interactively editable representation of the document and its org-mode representation. Leisure documents are served statically from a web server (but can persist changes by connecting to a local Emacs buffer on a client's machine). Leisure supports WebSocket-based remote collaboration over a separate relay server.

Eve is an ambitious project that wants to rethink programming for everyone. It introduces a new programming language in which all of the system state (including the UI) is addressable through queries. Eve takes a literate programming approach to developing full web-applications and uses an interactive notebook style user interface. Currently, the Eve project is still in development and (while planned) has yet to implement (real-time) collaboration and distribution across devices.

Codestrates combines conceptual ideas and technical implementations of these related works: it adopts an image-based persistence model inspired by Smalltalk, where the image is the content of a webpage. Similar to Lively, *Codestrates* builds

Interface Software and Technology. UIST '16. Tokyo, Japan: ACM, pp. 483–496. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984551. URL: http://doi.acm.org/10.1145/2984511.2984551.

[22] http://jupyter.org (last accessed: July 11, 2017)

[23] http://zeppelin.apache.org (last accessed: July 11, 2017)

[24] https://github.com/zot/Leisure (last accessed: July 11, 2017)

[25] http://witheve.com (last accessed: July 11, 2017)

[26] Eric Schulte and Dan Davison (2011). 'Active documents with org-mode.' In: *Computing in Science & Engineering* 13.3, pp. 66–73.

on modern web-technology, but does not abstract away from the DOM and conventional web development. *Codestrates* follows the literate computing approach (and visual structure) of interactive notebooks and the block-like code representation of online programming playgrounds, but it goes beyond in-line computation to programming and reprogramming applications—including itself. *Codestrates* provides Google Docs style real-time collaboration, but (by leveraging the Webstrates platform) shares the entire webpage instead of just the editor buffer. Finally, *Codestrates* adopts the prototype based approach to re-purposing software made by others from HyperCard.

9.3 CODESTRATES OVERVIEW

A codestrate is essentially a webpage whose content, presentation, and behaviour can be (collaboratively) edited from within the page and whose edits are inherently made persistent. It is automatically versioned and versions can be tagged with human-readable names. A codestrate is created by copying another codestrate and includes everything it needs to both implement and edit itself; and can be broken into three components: *paragraphs*, *sections* of related paragraphs, and the entire codestrate implementation (details are in the implementation section).

9.3.1 *Use of paragraphs and sections*

Paragraphs and *sections* in a codestrate are structured in a linear fashion, similar to traditional text documents. Figure 33 shows a schematic overview of the structure of a codestrate.

9.3.1.1 *Paragraph types and their function*

A *paragraph* can be of the type body, code, style, or data:

- Body paragraphs contain what is typically considered the content of a webpage and are directly editable through a simple rich-text editing interface or an HTML inspector (as illustrated in fig. 34).
- Code paragraphs contain editable JavaScript code that can be toggled to run on page load or be executed by pressing an execute button. The code in *Codestrates* executes in the run-time of the browser. Code paragraphs have syntax highlighting, identation, auto-completion, and an expandable interactive console for debugging.
- Style paragraphs contain Cascading Style Sheet (CSS) rules. Changes are immediately reflected on the page. They have syntax highlighting, identation, and auto-completion.
- Data paragraphs contain editable data in the JavaScript Object Notation (JSON) format; and they have syntax highlighting and identation.

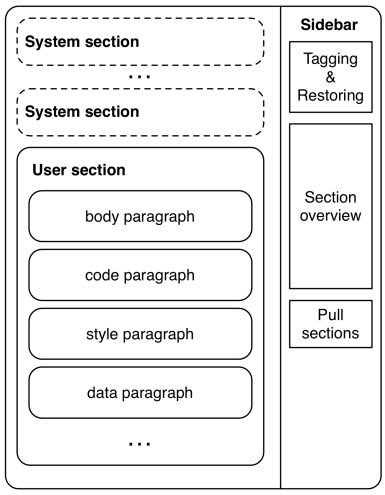


Figure 33. The structure of a codestrate. On the left are the sections, which include system sections (hidden by default) and one or more user sections. Sections can include paragraphs of different types (body, code, style, data). On the right is a sidebar (hidden by default), which contains actions for the codestrate (tag and restore, pull from another codestrate) and sections (toggle sections' visibility, add section).

Each paragraph can be expanded to full-screen, collapsed to a header only, locked against edits, deleted, and moved up or down in the list of paragraphs or across sections. They can also have a title and are addressable in JavaScript or in CSS through their (optional) IDs or classes.

In *Codestrates*, the result of evaluating a code paragraph does not have a standard output. Instead, the idea is to output evaluated code results into body paragraphs. To facilitate this, a body paragraph can include variables whose value can be set from code paragraphs through a simple API (more details in the implementation section).

9.3.1.2 *Section as paragraph collection*

Sections are collections of related paragraphs. We distinguish between *system* sections and *user* sections. System sections contain the implementation of the codestrate itself and are hidden by default. User sections contain whatever the user is working on. However, whether a section belongs to the user or the system is not fixed. Extending a codestrate with new functionality is essentially turning a user section into a system section by ticking a checkbox in the section's header.

All sections are listed in the sidebar, which also provides access to functions such as creating a new section, toggling a section's visibility, pulling sections from another codestrate, and tagging and restoring the codestrate.

The traditional boundaries between development and use are thus reduced in *Codestrates*. The user can fluidly move between the two, and to the extent that developing and using interactive systems are no longer necessarily separate activities. Figure 35 illustrates how a grocery list app can be developed in a codestrate with the UI expressed in a body paragraph and a style paragraph, and the inter-

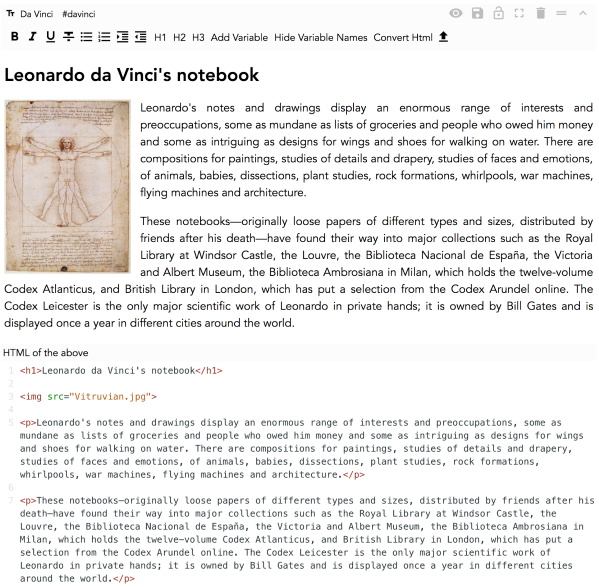


Figure 34. A codestrate in a *light* theme. It shows a body paragraph with its HTML inspector visible—visibility is toggled through the eye icon in the paragraph's header.

action implemented in a code paragraph that is set to run on page load (green running man). Full-screening the body paragraph turns the codestrate into a "regular" application, usable across devices (since the state is synchronized through the DOM). If changes to the application are required, the body paragraph can be minimized again and edits to the code can be made—collaboratively between multiple users, and in real-time.

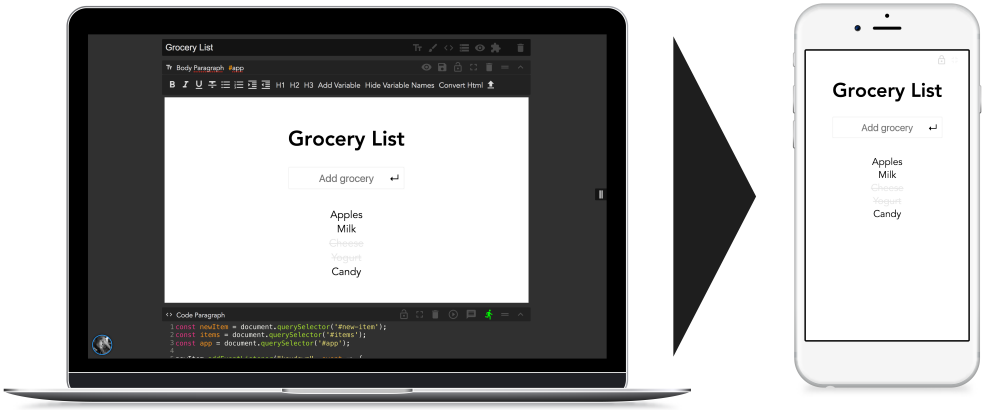


Figure 35. A simple grocery list implemented in a Codestrate. On the left, the codestrate in a desktop browser showing a body paragraph and the top of a code paragraph. To the right, the body paragraph has been made full-screen and loaded on a smartphone, now functioning as a grocery list app.

9.3.2 *Uses of Codestrates*

We present three scenarios that highlight *Codestrates*' capabilities, inspired by our own daily use over the course of six months. The scenarios assume a future of teachers and students fluent in computational thinking[27] who are able to master a medium that requires more technical knowledge than what is common today. The uses cases are also demonstrated in the supplementary video.

9.3.3 *Interactive notebooks in Codestrates*

Alex, a secondary school physics teacher prepares an assignment on speed and acceleration. He provides the students with a notebook for the assignment, including interactive code samples. One of these samples show how students can access the accelerometer and GPS data from their mobile phones (fig. 32A). Later and on their laptops, the students add a button that uses Alex' code sample to store the current position and acceleration in a data paragraph. They then open the codestrate on a smartphone while they walk, run, and cycle outdoors. When they return, they plot a graph of the captured data for another assignment in the notebook.

9.3.3.1 *How it works*

The assignment notebook is created by copying a codestrate, adding a section and a body paragraph to it, and using the rich-text capabilities to add content. Interfacing with the sensors on the device is done by writing JavaScript in a code paragraph and using the geolocation[28] and devicemotion[29] APIs. The sensors' current values are displayed through variables, which are inserted into a body paragraph and set through JavaScript in a code paragraph. The newest version of the codestrate is then tagged as stable and shared as a URL.

Users can create a personal copy of the stable version using the supplied URL with an additional `/?copy` query parameter. A data paragraph is added to the copy and the existing code paragraph is edited to store sensor values to the data paragraph. The HTML editor of a body paragraph can be used to add a button; and an additional code paragraph is added to make the button interactive. To plot data, the codestrate functionality needs to be extended either by writing code for custom visualization or by importing data visualization libraries (e.g. Vega Lite[30][31]). The implementation section contains details on how to import libraries.

[27] Wing, 'Computational Thinking.'
[28] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation (last accessed: July 11, 2017)
[29] https://developer.mozilla.org/en-US/docs/Web/Events/devicemotion (last accessed: July 11, 2017)
[30] https://vega.github.io (last accessed: July 11, 2017)
[31] Arvind Satyanarayan et al. (2017). 'Vega-lite: A grammar of interactive graphics.' In: *IEEE Transactions on Visualization and Computer Graphics* 23.1, pp. 341–350.

9.3.3.2 *In real use*

We have actively used *Codestrates* as an interactive assignment environment in our own classes. Twenty-five students in an introduction to programming course for non-computer science students completed their hand-ins for five consecutive weeks using *Codestrates*. We prepared assignments as codestrates with written instructions, interactive examples, automated testing, and code scaffolds (i.e., working but with incomplete code the students needed to edit). The assignment codestrates had a low threshold[32] and allowed our students to immediately start programming instead of fiddling with server settings, file uploads, or installing programming IDEs. We received positive feedback from the students through a dedicated feedback section in the codestrates they were using.

9.3.4 *Extending codestrates in Codestrates*

Alex notices that his students are collaborating remotely on their assignments after class. To hone his programming skills, he decides to extend the assignment codestrate with video communication. He copies a new codestrate and starts tinkering with *Web Real-Time Communications* (WebRTC). After a few iterations, Alex manages to stream audio and video from his web camera to all clients who have the same codestrate open (illustrated in fig. 32B). The codestrate already shows an avatar per connected client in the bottom left corner of the screen, and Alex overlays users' avatars to display their video streams instead. He pulls the section with the WebRTC code into the assignments codestrate and emails the students to let them know they can update their codestrates if they want to use this new functionality.

9.3.4.1 *How it works*

For video and audio streams, users can exploit the getUserMedia[33] API. The video and audio stream can be added to a DOM node with Webstrates' signal streaming API[34]. To build the UI of this new functionality, the user can add HTML, CSS, and JavaScript in body, style, and code paragraphs. The video element can be placed on top of the already existing avatar elements. The user can tag the current version of the codestrate with a meaningful name through an action in the sidebar. If an existing tag is reused, it will be updated to the current version. Other codestrates can update their sections with the changes using the "pull sections" function.

[32] Brad Myers, Scott E. Hudson, and Randy Pausch (Mar. 2000). 'Past, Present, and Future of User Interface Software Tools.' In: *ACM Trans. Comput.-Hum. Interact.* 7.1, pp. 3–28. ISSN: 1073-0516. DOI: 10.1145/344949.344959. URL: http://doi.acm.org/10.1145/344949.344959.

[33] https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia (last accessed: July 11, 2017)

[34] https://github.com/Webstrates/Webstrates (last accessed: July 11, 2017)

9.3.4.2 *In real use*

We have implemented several features by copying codestrates and pulling sections with new features back into the master codestrate. We usually started features in user sections but eventually changed them to system sections (e.g., remote pointers, video communication). The fluid way with which we can move between using and developing a codestrate results in the continuous development of functionality in response to specific tasks at hand. If the same functionality in one codestrate is needed at a later point in time for different tasks, those sections can easily be transferred between codestrates. For example, one user built a presentation tool for a research talk. That tool was then copied by someone else and extended with an in-slide code editor to teach a programming course. A static PDF viewer was adapted to a mobile note-taking tool that allowed hand-written annotations from an iPad, before being extended again into a review-writing tool with a simple text processor next to it. Through using/developing codestrates this way, we have access to an organically growing repository of functionality instead of a limited collection defined during the traditional development phase.

9.3.5 *Developing applications in Codestrates*

Jim and Bethany, two of Alex' students, are part of an extra-curricular game development club. They have been working on a multi-player tank game in a codestrate with the help of an open source web-based game engine. One day, while playing the game across their networked computers, they realize something is wrong with the physics of the bullets bouncing off of the walls. They exit the full-screen mode of the body paragraph that hosts the game view, and together edit the function that calculates the bullet path. The changes are immediately reflected in the running game, helping them iterate through different equations until they are satisfied (illustrated in fig. 32C).

9.3.5.1 *How it works*

Web-based game engines such as Phaser[35] can be used to render the game to a canvas element in a body paragraph. The game itself can be built using code paragraphs and is similar to regular game development in JavaScript. To add a multi-player mode in which the game state is synchronized between multiple players, users can—similar to the video streaming—leverage Webstrates' signalling API to send messages between clients of the same codestrate. In the game loop, it is possible to require the code of another code paragraph, which will allow for changing the game mechanics at run-time without reloading the page. Requiring code is explained in the implementation section.

9.3.5.2 *In real use*

We hired a professional game developer for a day to demonstrate it was possible to build a multi-player game in a codestrate. He implemented a simplified version of

[35] https://phaser.io (last accessed: July 11, 2017)

an existing tank game he had developed[36]. With only a brief introduction to how *Codestrates* works and the signalling API of Webstrates, he was able to implement the game without significant assistance over the course of a day. He struggled with the lack of screen real estate to get an overview of all his code, but enjoyed the ability to tweak gameplay mechanics and see them reflected in the game immediately. We will address this in the discussion section.

9.4 IMPLEMENTATION

Codestrates is implemented on top of Webstrates.[37] In order to understand the architectural choices and implementation details of *Codestrates*, it is necessary to understand how Webstrates works.

9.4.1 *How Webstrates works*

Webstrates consists of a web-server that persists all client-side DOM changes and synchronizes those changes to all clients of the same page. When a browser requests a page from the Webstrates server (e.g. /myWebstrate), it is served a generic HTML page containing a Webstrates client written in JavaScript. The client connects to the server through a web socket and receives the requested webstrate (e.g. myWebstrate) in a serialized JSON format. The client deserializes the JSON, populates the DOM of the generic HTML page, observes the DOM for changes using a `MutationObserver`, and listens on the web socket connection for changes made by other clients. Synchronization happens through operational transformation (OT)[38] using the open source OT framework ShareDB[39]. Webstrates uses OT to maintain a consistent document state across clients, consequently providing real-time collaborative editing of the DOM. Because ShareDB synchronizes operations on JSON documents, Webstrates' inner representation of the DOM is JSON using JsonML[40]. Webstrates leverages the principle of transclusion (using *iframes*) as a composition mechanism, which creates a dynamic document-application like relationship between two or more webstrates.

A new webstrate is created by requesting a webstrate that does not exist or by creating a copy of an existing webstrate. It can be copied either to a new webstrate with a random id (using /myWebstrate/?copy) or a named webstrate (using /myWebstrate/?copy=myWebstrateCopy).

Webstrates has a simple versioning mechanism based on logs of operations. For example, requesting /myWebstrate/1432 will retrieve the HTML of the 1432nd version of myWebstrate, and /myWebstrate/?restore=1432 will restore it to the state it had at the 1432nd version by applying the operations matching the difference between the current and the 1432nd version. Versions can be tagged with human-readable names and tags can be retrieved in the same manner as ver-

[36] https://www.tanktrouble.com/ (last accessed July 11, 2017)
[37] Klokmose et al., 'Webstrates: shareable dynamic media.'
[38] Clarence A Ellis and Simon J Gibbs (1989). 'Concurrency control in groupware systems.' In: *Acm Sigmod Record*. Vol. 18. 2. ACM, pp. 399–407.
[39] https://github.com/share/sharedb (last accessed: July 11, 2017)
[40] http://www.jsonml.org (last accessed: July 11, 2017)

sions. The Webstrates server automatically generates a tag for a webstrate when no edits were made for a set period of time. Webstrates uses external authentication providers (e.g. GitHub). User rights such as read, read-write, or none can be added to a webstrate using the `data-auth` attribute on the `html` element. User information such as username and a user's avatar are accessible through a JavaScript API.

9.4.1.1 *Extensions to Webstrates*

In parallel to *Codestrates*, Webstrates has been extended with a number of features that makes the development of larger systems more convenient. *Codestrates* relies on many of these extensions, e.g., tagging, transient elements, assets, signalling, and a WebRTC API. A `transient` element has been introduced that allows for adding elements to the DOM that are not persisted and synchronized to other clients. A custom context menu is an example of something that makes sense to put in a transient element as the context menu is ephemeral and only relevant for the user who opened it. Binary assets such as images or videos can be attached to a webstrate (through a POST request) and accessed using e.g. myWebstrate/myVideo.mp4. Assets are versioned like any other change made to the webstrate document. A signalling API has been developed to allow clients of the same webstrate to communicate with each other without manipulating the DOM. Clients can broadcast signals to every client of the same webstrate or send targeted signals to a subset of clients (client list accessible through the API). Signals can contain JSON objects as message payloads. For example, *Codestrates* uses signals to communicate ephemeral states such as remote cursor and pointer positions, or to establish WebRTC connections between clients of the same codestrate. Finally, signal streaming allows peer-to-peer communication between clients using WebRTC. *Codestrates* uses signal streaming for video+audio communication.

9.4.2 *Codestrates*

Every codestrate is a webstrate that includes the codestrate implementation and the user content: it is completely self-contained. The markup (HTML), styling (CSS), program code (JavaScript), and data (JSON) are stored in a codestrate's document body, each wrapped in a paragraph element (`<div class="paragraph">`). As a structuring mechanism, paragraphs are grouped inside sections (`<div class="section">`). Webstrates ensures that changes to the content of a codestrate are made persistent and synchronized to all clients of the same codestrate. Listing 1 shows a simplified HTML of the document structure with two system sections (one hidden), a user section, three code paragraphs (one hidden by its section), a body paragraph, a style paragraph, and a data paragraph.

```
1 <html>
2  <head>
3   <script type="text/javascript">
4    <!-- see Listing 2 -->
5   </script>
6  </head>
7  <body>
8   <div class="section section-hidden" data-type="system">
```

```
9     <div id="bootstrap" class="paragraph code-paragraph">
10     <pre type="text/javascript">
11      // Code that queries and executes all
12      // run-on-load code paragraphs
13     </pre>
14    </div>
15   </div>
16   <div class="section" name="A System Section"
17        data-type="system">
18    <div class="paragraph code-paragraph"
19         run-on-load="true">
20     <pre id="code-on-load" type="text/javascript">
21      // Executed by code paragraph in line 10
22     </pre>
23    </div>
24   </div>
25   <div class="section" name="A User Section">
26    <div class="paragraph body-paragraph">
27     <div id="body" class="class1 class2"
28          contenteditable="true">
29      <!-- HTML -->
30     </div>
31    </div>
32    <div class="paragraph style-paragraph">
33     <style id="style" type="text/css">
34      /* CSS */
35     </style>
36    </div>
37    <div class="paragraph code-paragraph">
38     <pre id="code" type="text/javascript">
39      // JavaScript
40     </pre>
41    </div>
42    <div class="paragraph data-paragraph">
43     <pre id="data" type="application/json">
44      /* JSON */
45     </pre>
46    </div>
47   </div>
48  </body>
49 </html>
```

Listing 1: Simplified HTML structure of a codestrate.

9.4.2.1 *Bootstrapping and code execution*

Code execution in *Codestrates* differs from the regular JavaScript execution routine of a browser. Its execution relies on three integral conditions: (i) there is bootstrap code in a `script` element in the document `head` (Listing 1, line 3); (ii), all other code is stored in `pre` elements, which are not executed on page load; and (iii) one code paragraph has the id `#bootstrap` (Listing 1, line 9).

To open a codestrate, the four lines of bootstrap code in (Listing 2) are executed after the webstrate is loaded. This triggers the execution of the code paragraph with the id `#bootstrap`, which queries the DOM for all other code paragraphs with the attribute `run-on-load` set to `true` (e.g., Listing 1, line 19) and synchronously executes them in the order as they appear in the DOM.

```
1 webstrate.on("loaded", function () {
2  var codeParagraph = document.querySelector("#bootstrap");
3  new Function(codeParagraph.textContent)();
```

```
4 });
```

Listing 2: *Codestrates*' bootstrap JavaScript code.

Each code paragraph is executed in its own scope and execution context using the `Function` object. Code paragraphs do not create closures to their creation contexts and are only able to access their own local variables and variables defined in global scope (e.g., document object). This prevents users from accidentally overriding and interfering with *Codestrates*' execution logic. The execution context provides access to a proxied console and a `Variable` object. Each code paragraph has its own console output; all `log`, `error`, `debug`, `warn`, and `info` function calls on the console object are logged to this output before they are redirected to the window's console. The `Variable` object provides convenience functions to replace content in a body paragraph. For example, a variable *myVar* is inserted into a body paragraph using its rich-text editor tools (see *Add Variable* in fig. 36). The content of *myVar* can be set from any code paragraph using `Variable("myVar").set("newValue")`. The variable is represented as `<div class="variable" data-name="myVar"></div>` in the body paragraph.

9.4.2.2 *Requiring modules and importing external libraries*

A code paragraph can require another code paragraph using CSS selectors (e.g., `var myModule = require("#myModule")`) or class selectors (e.g. `require(".myModules")`). When code is required using a class selectors, all code paragraphs of that particular class are queried, their contents are concatenated, and the code is executed as one script. Executing the content of a required code paragraph adds an additional `exports` object in the execution context. An imported code paragraph can export variables and functions using the exports object (e.g., `exports.myVar = "myValue"` or `exports.myFunction = function() {...}`). The exports object is a plain key value store that allows code paragraphs to export multiple variables and functions. The require function will return the exports object in order for the caller code paragraph to access exported variables and functions (e.g., `myModule.myVar` or `myModule.myFunction()`).

External JavaScript libraries can be used through `importLib` (Listing 3), which takes either a single URL or reference to a webstrate, or an array of them. For each URL, *Codestrates* adds a `transient` element in the document `head` with the `script` element inside. This way, added scripts do not persist and are not synchronized with other clients. Because browsers execute `script` elements asynchronously when added at runtime, two external JavaScript libraries that depend on each other can cause faulty code when added after page load. *Codestrates*' `importLib` guarantees that external JavaScript libraries are loaded and executed synchronously in the order in which they have been defined.

The `importLib` function returns a promise which resolves after all libraries have been imported (i.e., loaded and executed). The code in the resolve function can then use imports the same way as `script` elements that are loaded synchronously. Importantly, subsequent code paragraphs will be blocked until the preceding code paragraph has *imported* and executed all external libraries, executed all code paragraphs that are *required* within, and finally executed all of its own code.

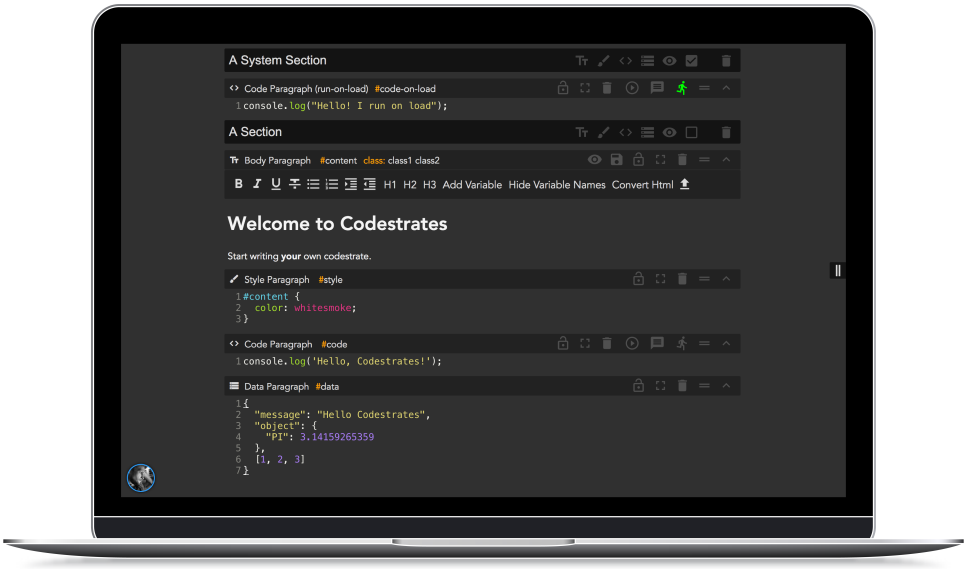


Figure 36. Codestrate view of Listing 1, including paragraphs and their contents. The section containing the bootstrap code is hidden.

```
importLib([
  "//cdn/extLibrary.js",
  "assetLibraryDependingOnExtLibrary.js"
]).then(() => {
  // code executes after both libraries are imported
});
```

Listing 3: Import external libraries in a *Codestrates* code paragraph.

9.4.2.3 *Generating user interfaces and structuring paragraphs' data*

As part of *Codestrates*' core system functionality, the user interface elements for sections and paragraphs are generated at runtime (e.g., the rich-text editor tools for a body paragraph (fig. 36)). The UI elements are programatically created using `transient` elements in order to keep their states local instead of synchronized between clients. This way, expanding the HTML editor of a body paragraph only has effect locally. A `MutationObserver` observes the `body` element of a codestrate and its subtree to create user interfaces for sections and paragraphs that are added to the document after initial loading.

The content of body paragraphs are contenteditable `div` elements directly visible to the user. All changes within a `div` element are immediately reflected in the DOM and Webstrates synchronizes changes with other users of the same codestrate.

For the code, style, and data paragraphs, the element storing their content is hidden from the users through CSS, and a `transient` element is generated with a CodeMirror[41] based editor that creates a two-way binding between its text buffer and (remote) changes to the element's content. CodeMirror comes with

[41] http://codemirror.net (last accessed: July 11, 2017)

support for syntax highlighting, code completion, formatting, and other functionality expected from modern code editors. As an optimization we only instantiate CodeMirror instances for visible paragraphs.

CSS is stored directly in a `style` element. Therefore, changes to CSS rules in style paragraphs have an immediate effect on the rendering of the codestrate's content. Code is executed either explicitly by the user through pressing the execute button in the code paragraph's toolbar, or after the webstrate loaded if the user enabled the run-on-load in the code paragraphs toolbar (the green running man fig. 36, top).

9.4.2.4 *Remote collaboration*

As Webstrates synchronizes the DOM between connected clients transparently, *Codestrates* allows for remote collaboration. However, ephemeral data such as pointers (e.g., mouse cursor or touch points) are not synchronized.

Using signalling, we implemented a user manager and remote pointers in *Codestrates* to support awareness of other users. Every client that joins a codestrate broadcasts the user's information to all other clients. If a user is authenticated, the codestrate broadcasts username, friendly name, and the avatar url of the user; otherwise it broadcasts anonymous with a default avatar. The default codestrate UI shows all connected users as avatars in the bottom left corner in a `transient` element. Each user gets assigned a distinct color that is visible as the border of their avatar.

Each client broadcasts pointer positions (such as mousemove and touchmove) and pointer actions (such as click and tap). Remote pointers are represented as `transient` elements positioned relative to the nearest paragraph, which share dimensions across devices and screen sizes. Actions are styled `div` elements appended to the pointer element and removed after a timeout. Cursor positions and selections in editors are synchronized across clients to allow for Google Docs-like document-centric authoring. The user color from the user manager is used to color pointers and cursors to associate actions to remote users.

Video and audio communication is implemented using Webstrates' `transient` element and signal streams, and the `getUserMedia` API. A codestrate listens for incoming streams from other clients, automatically accepts them, and overlays the sender's avatar with a `video` element with the stream. The `video` element is wrapped in a `transient` element and not synchronized with other clients. Generally, a user can exploit Webstrates' `transient` element in a codestrate to write an application that has a visually distinct appearance on different clients or different users. To start video and audio streaming, a user has to click on the videocamera icon that is revealed when hovering over the user's own avatar.

9.4.2.5 *Creating, versioning, and updating*

Codestrates provides a UI for easily tagging and restoring versions of a codestrate. Pressing the restore button in the sidebar shows a dialog with a list of both user-generated and auto-generated tags, which are accessed using the Webstrates versioning API.

Because a codestrate contains all of its implementation, it needs to be updated manually in case a new feature is introduced or a bug is found in the codestrate from which it was copied. To support this, users can pull sections from other codestrates or even from an earlier version of the same codestrate. The pull sections functionality loads a codestrate of a specific version into the caller codestrate using an `iframe` wrapped in a `transient` element. To pull a section, the user provides the codestrate id and (optionally) specifies the version of the codestrate to be pulled and a CSS selector for particular elements of the codestrate (if unspecified, the latest version and the "system section" selector will be used). The codestrate then deletes all of its sections that match the CSS selector; queries sections matching the CSS selector in the codestrate being pulled and deep clones them; appends the cloned elements to itself; and finally reloads the webpage after all operations (OT) are synchronized with the server. While pulling sections would work without a reload, doing so ensures code written by others always runs without side-effects.

9.5 DISCUSSION

9.5.1 *Limitations and future work*

9.5.1.1 *Out-of-browser code exection and Jupyter integration*

Code execution in *Codestrates* happens at run-time and only within the browser which limits language support to JavaScript (or languages compiled or transpiled to it) and means it does not have access to the operating system of the host computer. However, we have successfully experimented with using a Jupyter kernel running on the local machine to execute Python code from a codestrate. We created Python specific code paragraphs that post their code over HTTP to the local Jupyter kernel when executed and have the standard output redirected to the console of the code paragraph. Future research includes a model for managing in-browser and out-of-browser computation (e.g. on a local computer or an online service) and polyglot code-execution in a codestrate where various programming languages can be combined to perform a series of dependent computations.

9.5.1.2 *Version control*

Version control systems (e.g., Git or Subversion) are essential for systems development. *Codestrates* allows tagging of and branching from codestrates, but updating from another codestrate replaces affected paragraphs entirely. It does not support automatic or manual merging of paragraph contents. For future work we plan to integrate merging algorithms, e.g. a recursive three-way merge.

Codestrates "pull section" mechanism allows for very idiosyncratic ways of feature and application development: (i) feature development for *Codestrates* can happen in a copy of the *Codestrates* prototype and after testing be pulled back into the prototype, (ii) copies can be updated to the latest version of the *Codestrates* prototype by pulling the latest version of system sections, and (iii) (sys-

tem) sections in a codestrate can be downgraded without *undoing* the changes in-between, e.g., changes that happened to other sections.

9.5.1.3 *Overhead of development environment*

Applications developed with *Codestrates* carry more weight than regular web applications because they include both the application code and a development environment. Loading time and memory consumption of an application built with *Codestrates* is higher than a comparable traditional web application. The overhead adds ~800 kilobytes of data (including external and non-minified libraries like CodeMirror). Future work includes caching strategies for external libraries and lazy loading of the development environment.

9.5.1.4 *Textual programming*

Codestrates currently only allows users to express interactive behavior through textual programming using JavaScript and the DOM API. Visual languages such as Scratch or languages influenced by natural language such as HyperTalk have successfully introduced beginners and children to programming. These approaches could be integrated in *Codestrates* and even be used interchangeably, so that *Codestrates* can adapt to the experience level of the programmer. We strongly believe that the threshold for expressing interaction and computation in a codestrate should be lower so it can be used by a wider audience with different levels of computational literacy.[42]

9.5.1.5 *Usability*

Full-screening a paragraph is currently not a transient action, which means that it affects all clients of the same codestrate. This makes it impossible for a user to change the style or code of a codestrate while also running it as a full-screen application on another device or for multiple users to view the codestrate in different ways. Adding this flexibility would mean users could maximize the use of screen real-estate e.g. across multiple screens instead of continuously scrolling through the document, as mentioned as a limitation by the game developer who implemented the multi-player game in *Codestrates*. This would be possible by CSS rules targeting transient attributes available with the newest version of Webstrates; attributes that are not persisted and synchronized.

While the CodeMirror library we use in *Codestrates* provides many of the editing capabilities of modern programming IDEs, it is still limited compared to desktop editors. For example, it lacks advanced auto-completion and refactoring features now taken for granted when developing software (and their absence was noticed by our students).

[42] Myers, Hudson, and Pausch, 'Past, Present, and Future of User Interface Software Tools.'

9.5.1.6 *Limitations from Webstrates*

Codestrates inherits the limitation from Webstrates that fine-grained permissions beyond a per-document permission model (no access, read-only, or read-and-write) is a significant challenge.[43]

9.5.2 *Systems-oriented evaluation*

According to Olsen, a user interface system can be evaluated according to its "solution viscosity", which expresses the effort of a programmer to create possible solutions. Good systems can reduce solution viscosity in three ways: *flexibility*, *expressive leverage*, and *expressive match*.

9.5.2.1 *Flexibility*

Good flexibility allows the user to make rapid changes and evaluate them immediately.[44] *Codestrates* supports flexibility with paragraphs and sections that remain continuously inspectable and whose changes can be evaluated at runtime. For example, a user can customize the look of a codestrate-based application by changing CSS properties in a style paragraph, change application behavior by changing the respective code paragraph, or add additional functionality by writing new code.

9.5.2.2 *Expressive leverage*

A system with a high expressive leverage reduces the choices a user can make while still being able to express more.[45] Since *Codestrates* builds on Webstrates, all content is by default persisted and synchronized. As a result, there are no additional software layers necessary to add real-time collaboration and there is no need to create databases or add service to persist application data.

9.5.2.3 *Expressive match*

The expressive match refers to how close "the means for expressing design choices are to the problem being solved".[46] With *Codestrates*, we anticipate a future generation where computational thinking is part of formal education practices and considered a "fundamental skill for everyone".[47] Although this generation will know how to express computation in code, they are considered non-professional programmers[48] and may not have acquired a fundamental knowledge about computing technology (e.g., an understanding of client/server communication). Olsen

[43] Klokmose et al., 'Webstrates: shareable dynamic media.'
[44] Olsen, 'Evaluating User Interface Systems Research.'
[45] Ibid.
[46] Ibid.
[47] Wing, 'Computational Thinking.'
[48] Margaret M. Burnett and Brad A. Myers (2014). 'Future of End-user Software Engineering: Beyond the Silos.' In: *Proceedings of the on Future of Software Engineering*. FOSE 2014. Hyderabad, India: ACM, pp. 201–211. ISBN: 978-1-4503-2865-4. DOI: 10.1145/2593882.2593896. URL: http://doi.acm.org/10.1145/2593882.2593896.

argues that new tools should be "accessible, easier or more effective for this desired population" and they should support "different norms of expression or design goals that are not supported by existing tools." With literate computing expressed in *Codestrates*' paragraphs, users can "read a program" from top to bottom to understand its execution order, much like reading a book or an article. This is greatly different from often complex dependencies in file-based projects. It is also different from a single HTML file containing HTML, JavaScript, and CSS edited in a file editor; paragraphs containing HTML, JavaScript, or CSS are visually distinct from each other and provide additional tools like "execute" for code paragraphs or rich-text editor tools for body paragraphs. Changes to a body, style, code, or data paragraphs are immediately part of the codestrate that contains them and thus are already "deployed." There is no need to copy code from a playground, paste it to a file, and deploy it on a server to make it accessible to everyone. *Codestrates* is agnostic to any programming pattern, and might in the future even be agnostic to the programming language (based on ongoing work). Users can exploit their pre-existing knowledge on web development without the need to learn or adapt to a new programming language.

9.6 CONCLUSION

Most modern applications create a strict divide between the development of the software and the use of it. We have demonstrated how a literate computing approach can close that gap by allowing users to mix prose and computation in the same perceptual space. This not only supports the execution of programs within an application, but can also be used to extend and reprogram the functionality of an interactive notebook from within itself, resulting in an inherently negotiable computational medium.

Codestrates is built on Web standards and is easily appropriable and usable by anyone with a basic Web development background. However, the presented use cases in this paper are based on currently fictional levels of technical proficiency of non-professional programmers. Future research includes iteratively co-designing codestrates to support real users such as data scientists, secondary school teachers, and students; and exploring what could aid or prevent end-user adoption of a medium such as *Codestrates*.

10.

BETWEEN SCRIPTS AND APPLICATIONS: NEGOTIABLE SOFTWARE FOR THE FRONTIER OF NANOSCIENCE

Based on Nouwens, Borowski, Fog, and Klokmose[1]

10.1 INTRODUCTION

There has been a recent upsurge in commercial products and research prototypes that run counter to the application paradigm and instead look more like *computational media* — software which allows you to create, execute, and edit computations that can operate on anything, including the software itself. The computational notebooks that have become popular in data science, digital journalism, and education, for example (e.g., Jupyter Notebook[2] or Observable[3]), allow users to mix traditional word-processing functionality with executable code in the same environment.

As seeds for a paradigm shift of software, however, computational notebooks are programming environments with some application-like qualities, rather than GUI applications that also allow for writing and executing computations. As such, they treat code as the primary object of interest and expect users with software engineering skills (e.g., version control, dependency management). We want to expand this reimagination of software as computational media and explore designs that support computer users who are dependent on computation for their activities, but who are not trained as software engineers.

We collaborated with a group of biomolecular nanoscientists whose scientific practice and progress rely on using in-house developed computational tools, but who do not have a formal programming education. We engaged them in the participatory design of a *computational labbook* that helps them carry out a key part

[1] Midas Nouwens et al. (2020a). 'Between Scripts and Applications: Computational Media for the Frontier of Nanoscience.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376287. URL: https://doi.org/10.1145/3313831.3376287.

[2] Juypter Notebook (2019). URL: https://jupyter.org.

[3] Observable (2019). URL: https://observablehq.com.

of their work, namely the computational design of new RNA structures. The prototype and participatory design process were guided by the concepts *computability*, *malleability*, *shareability*, and *distributability*; four proposed principles for computational media derived from Klokmose et al.[4] and Rädle et al.[5]

We address the following research questions:

1. How do biomolecular nanoscientists use computation in their work?
2. How can computational media, rather than applications and scripts, better support their work?
3. What can we learn about computational media at large by designing for biomolecular nanoscientists?

Accordingly, our main contributions are: (i) an empirical account of the computational activities and challenges of biomolecular nanoscientists; (ii) a high-fidelity computational labbook prototype for the domain of biomolecular nanoscience; and (iii) a deepening of the four principles for computational media rooted in real-world work praxis.

10.2 RELATED WORK

We examine the potential and principles of computational media in a scientific practice. Related work includes computational tools in the sciences and work on computational media.

10.2.1 *Lab Notebooks and e-Science Tools*

In the sciences, the laboratory notebook is often seen as the cornerstone of research, and efforts to create electronic lab notebooks (ELNs) are plenty. Commercial ELNs are largely wiki-based and oriented towards particular considerations such as transparency, data management, and ease-of-use (e.g., Confluence,[6] BIOVIA,[7] and Labguru[8]). Some academic work has explored other aspects of electronic laboratory work. Inspired by the Semantic Web, Talbott et al.[9] (among others[10]) have focused on data provenance as a key factor in lab work. Others have

[4] **Klokmose2015**.

[5] Roman Rädle et al. (2017). 'Codestrates: Literate Computing with Webstrates.' In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST '17. Québec City, QC, Canada: ACM, pp. 715–725. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126642. URL: http://doi.acm.org/10.1145/3126594.3126642.

[6] Atlassian Confluence (2019). URL: https://atlassian.com/confluence.

[7] BIOVIA (2019). URL: https://3dsbiovia.com.

[8] Labguru (2019). URL: https://labguru.com.

[9] Tara Talbott et al. (2005). 'Adapting the Electronic Laboratory Notebook for the Semantic Era.' In: *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005.* Pp. 136–143. DOI: 10.1109/ISCST.2005.1553305.

[10] Zulkifly M. Zaki et al. (2011). 'A User-orientated Electronic Laboratory Notebook for Retrieval and Extraction of Provenance Information for EUROCHAMP-2.' In: *2011 IEEE Seventh International Conference on eScience*, pp. 371–378. DOI: 10.1109/eScience.2011.58; Zulkifly M. Zaki et al. (2013). 'Architecture design of a user-orientated electronic laboratory notebook: A case study within an atmospheric chemistry community.' In: *Future Generation Computer Systems* 29.8, pp. 2182 –2196. ISSN: 0167-739X. DOI: 10.1016/j.future.2013.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0167739X1300071X.

aimed to bridge the gap between physical and digital materials by providing an interactive tabletop in the lab space,[11] augmenting the work-space with mixed-reality "interactive paper",[12] or using digital photographs to couple notebooks and external materials.[13] Oleksik et al. have investigated the artifact ecologies of laboratories[14] and how scientists appropriate existing off-the-shelf note-taking software and the inherent clash between stability and flexibility, providing guidelines to how ELNs might be designed.[15]

Most of the work on ELNs, however, has sought to more or less replace physical notebooks with digitally enhanced replicas. These enhancements are often web-based and collaborative, offering tools for cross-referencing, tagging, and visualization.[16] While these are admirable efforts in their own right, we believe that looking more closely at the potential of computational media can fruitful directions for re-thinking laboratory notebooks to be the locus of scientific computational activities — as also proposed in.[17]

[11] Aurélien Tabard et al. (2011). 'The eLabBench: An Interactive Tabletop System for the Biology Laboratory.' In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS '11. Kobe, Japan: ACM, pp. 202–211. ISBN: 978-1-4503-0871-7. DOI: 10.1145/2076354.2076391. URL: http://doi.acm.org/10.1145/2076354.2076391.

[12] Wendy E. Mackay (2003). 'The Missing Link: Integrating Paper and Electronic Documents.' In: *Proceedings of the 15th Conference on L'Interaction Homme-Machine*. IHM '03. Caen, France: ACM, pp. 1–8. ISBN: 1-58113-803-2. DOI: 10.1145/1063669.1063671. URL: http://doi.acm.org/10.1145/1063669.1063671.

[13] Ron Yeh et al. (2006). 'ButterflyNet: A Mobile Capture and Access System for Field Biology Research.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: ACM, pp. 571–580. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124859. URL: http://doi.acm.org/10.1145/1124772.1124859.

[14] Gerard Oleksik, Natasa Milic-Frayling, and Rachel Jones (2012). 'Beyond Data Sharing: Artifact Ecology of a Collaborative Nanophotonics Research Centre.' In: *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*. CSCW '12. Seattle, Washington, USA: Association for Computing Machinery, 1165–1174. ISBN: 9781450310864. DOI: 10.1145/2145204.2145376. URL: https://doi.org/10.1145/2145204.2145376.

[15] Gerard Oleksik, Natasa Milic-Frayling, and Rachel Jones (2014). 'Study of Electronic Lab Notebook Design and Practices That Emerged in a Collaborative Scientific Environment.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: Association for Computing Machinery, 120–133. ISBN: 9781450325400. DOI: 10.1145/2531602.2531709. URL: https://doi.org/10.1145/2531602.2531709.

[16] Georgios John Fakas, Anh Vu Nguyen, and Denis Gillet (2005). 'The Electronic Laboratory Journal: A Collaborative and Cooperative Learning Environment for Web-Based Experimentation.' In: *Computer Supported Cooperative Work (CSCW)* 14.3, pp. 189–216. ISSN: 1573-7551. DOI: 10.1007/s10606-005-3272-3. URL: https://doi.org/10.1007/s10606-005-3272-3; Francois Roubert and Mark Perry (2013). 'Putting the Lab in the Lab Book: Supporting Coordination in Large, Multi-site Research.' In: *HCI 2013 - 27th International British Computer Society Human Computer Interaction Conference: The Internet of Things*. URL: http://dl.acm.org/citation.cfm?id=2578048.2578066; Jenifer L. Skidmore et al. (1998). 'A Prototype Notebook-based Environment for Computational Tools.' In: *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*. SC '98. San Jose, CA: IEEE Computer Society, pp. 1–15. ISBN: 0-89791-984-X. URL: http://dl.acm.org/citation.cfm?id=509058.509080; Phil Turner and Susan Turner (1997). 'Supporting Cooperative Working Using Shared Notebooks.' In: *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*. Dordrecht: Springer Netherlands, pp. 281–295. ISBN: 978-94-015-7372-6. DOI: 10.1007/978-94-015-7372-6_19. URL: https://doi.org/10.1007/978-94-015-7372-6_19.

[17] Clemens N. Klokmose and Pär-Ola Zander (2010). 'Rethinking Laboratory Notebooks.' In: *Proceedings of COOP 2010*. Springer, pp. 119–139. DOI: 10.1007/978-1-84996-211-7_8.

The natural sciences have been the targets of efforts to augment the scientific process with computational capabilities, in particular when their objects are large-scale and complex data, e.g., genomic data, which often is referred to as *e-Science*. Smith[18] argues that contemporary scientific work is increasingly reliant on software engineering abilities. Surveying the literature on e-science reveals a large body of work that focuses on the technical infrastructure for distributed and grid computing. However, there is also work that addresses challenges similar to some of those we observe in our studies: Holdgraf et al.[19] explore how to facilitate sharing of computational environments, Östberg et al.[20] study how to ease distribution of computations by raising the required level of abstraction, and Harrison et al.[21] developed a standard for sharing scientific computational workflows. However, their aim to create practical computational tools to support scientific work, rather than change the dominant software paradigm. We argue that many of these challenges are tied together and can be addressed by a change in focus from scripts and applications to computational media.

10.2.2 *Computational Media*

The idea of software as a dynamic or computational medium was popularised by Alan Kay in his Dynabook vision.[22] diSessa exemplified a computational medium[23] with his Boxer programming environment[24] that allowed users to build software and execute computation in the same medium. diSessa contrasted the "monolithic, nonmodifiable applications" with computational media, perceiving the latter as tools that can be organically enriched and extended[25] (fig. 37).

[18] Spencer Smith (2018). 'Beyond Software Carpentry.' In: *Proceedings of the International Workshop on Software Engineering for Science*. SE4Science '18. Gothenburg, Sweden: Association for Computing Machinery, 32–39. ISBN: 9781450357487. DOI: 10.1145/3194747.3194749. URL: https://doi.org/10.1145/3194747.3194749.

[19] Chris Holdgraf et al. (2017). 'Portable Learning Environments for Hands-On Computational Instruction: Using Container- and Cloud-Based Technology to Teach Data Science.' In: *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. PEARC17. New Orleans, LA, USA: Association for Computing Machinery. ISBN: 9781450352727. DOI: 10.1145/3093338.3093370. URL: https://doi.org/10.1145/3093338.3093370.

[20] P. Östberg et al. (2012). 'Reducing Complexity in Management of eScience Computations.' In: *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 845–852. DOI: 10.1109/CCGrid.2012.72.

[21] Andrew Harrison et al. (2011). 'Object Reuse and Exchange for Publishing and Sharing Workflows.' In: *Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science*. WORKS '11. Seattle, Washington, USA: Association for Computing Machinery, 67–76. ISBN: 9781450311007. DOI: 10.1145/2110497.2110506. URL: https://doi.org/10.1145/2110497.2110506.

[22] Alan C. Kay (1972). 'A Personal Computer for Children of All Ages.' In: *Proceedings of the ACM Annual Conference - Volume 1*. ACM '72. Boston, Massachusetts, USA: Association for Computing Machinery. ISBN: 9781450374910. DOI: 10.1145/800193.1971922. URL: https://doi.org/10.1145/800193.1971922.

[23] Andrea A. diSessa (2001). *Changing Minds: Computers, Learning, and Literacy*. Mit Press. ISBN: 9780262041805.

[24] Andrea diSessa and Hal Abelson (Sept. 1986). 'Boxer: A Reconstructible Computational Medium.' In: *Commun. ACM* 29.9, pp. 859–868. ISSN: 0001-0782. DOI: 10.1145/6592.6595. URL: https://doi.org/10.1145/6592.6595.

[25] diSessa, *Changing Minds: Computers, Learning, and Literacy*.

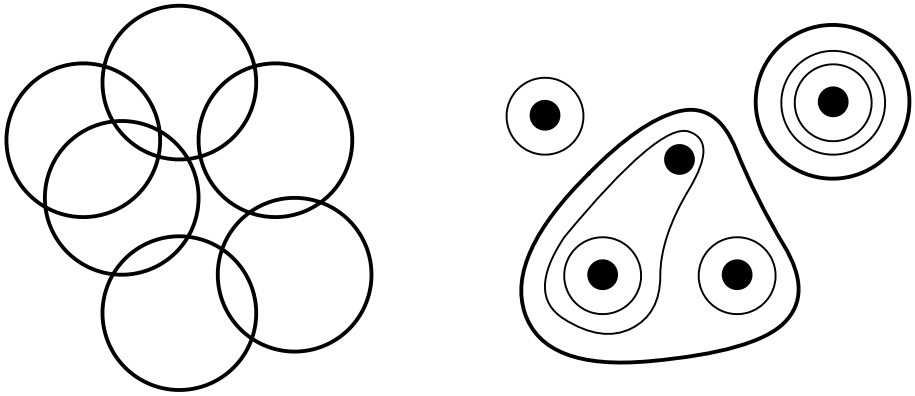


Figure 37. Applications (left) and computational media (right). Adapted from diSessa.[26]

In recent years, the proliferation of computational notebooks, e.g., Jupyter Notebook,[27] is emblematic of the perceived and actual benefits of computational media. In 2017, over a million Jupyter notebooks had been shared on GitHub.[28] Computational notebooks has has also recently become an academic object of study in HCI (e.g.,[29]). Computational notebooks are not ELNs by another name, rather they are computational environments that enables weaving "a narrative directly into a live computation, interleaving text with code and results to construct a complete piece that relies equally on the textual explanations and the computational components".[30] They are particularly designed to support replication in computational sciences.[31] Computational media do not necessitate that the user becomes a programmer. A particular group of these media can be termed *no-code computational media*. No-code computational media embody computational capabilities but do not require that the users themselves write program code. One such example is Notion,[32] a knowledge repository environment that enables users to utilise computation without writing code. Another example is Coda.[33]

[27] Juypter Notebook,

[28] Adam Rule, Aurélien Tabard, and James D. Hollan (2018). 'Exploration and Explanation in Computational Notebooks.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 32. DOI: 10.1145/3173574.3173606.

[29] Andrew Head et al. (2019). 'Managing Messes in Computational Notebooks.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, p. 270. DOI: 10.1145/3290605.3300500; Mary Beth Kery and Brad A. Myers (2018). 'Interactions for Untangling Messy History in a Computational Notebook.' In: *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, pp. 147–155. DOI: 10.1109/VLHCC.2018.8506576; Mary Beth Kery et al. (2018). 'The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 174. DOI: 10.1145/3173574.3173748; Adam Rule et al. (2018). 'Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding.' In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW, p. 150. DOI: 10.1145/3274419; Rule, Tabard, and Hollan, 'Exploration and Explanation in Computational Notebooks.'

[30] Fernando Pérez and Brian Granger (2015). *Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science*. URL: https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58.

[31] Kluyver et al., 'Jupyter Notebooks–a publishing format for reproducible computational workflows.'

[32] Notion (2019). URL: https://notion.so.

[33] Coda (2019). URL: https://coda.io.

Webstrates[34] is a platform for developing dynamic or computational media inspired by Kay's early visions of personal dynamic media. Webstrates is based on a simple change to the mechanics of the web, where a web-page is made collaboratively editable directly from the browser. In Webstrates there is no technical distinction between editing content and the scripts defining the functionality of a page. This means that the traditionally hard distinction between development and use of software is blurred. Codestrates[35] is an authoring environment for Webstrates and builds on the computational notebook metaphor. However, it allows for the development of application-like collaborative software, and is reprogrammable and extensible from within. Webstrates and Codestrates are document-centric in the spirit of Boxer.

Webstrates is based on the principles of shareability, distributability, and malleability. *Shareability* refers to the users' ability to share and collaborate seamlessly on multiple types of data within a document, synchronously and asynchronously, with themselves and multiple people, using their own personalised views and tools. *Distributability* points to the ability to easily distribute documents, interactive elements, and computation across multiple and heterogeneous devices and platforms. *Malleability* signifies the system's ability to be continuously changed and appropriated by users in personal and idiosyncratic ways. We used these principles as guiding design principles for our participatory design process with the inclusion of the principle of *computability*, which is left implicit in previous work on computational media. With *computability*, we refer to the ability to develop, edit, and execute simple and complex computations on data that exist in the same perceptual space as the code. We used Webstrates and Codestrates to create our prototype.

10.3 METHOD

Our aim was to explore computational media as an alternative vision of software to traditional applications and scripts. To make this actionable, we looked for use cases where computation was an integral part of users' activities, but where users were not employed or trained as a programmer; activities where using and developing a piece of software blur, and users who work close to the code but did not necessarily author it.

We collaborated with a group of biomolecular nanoscientists because they (a) do complex knowledge work mediated by digital tools and materials, some of which are developed by lab themselves; (b) are computationally dependent but not trained to be computationally literate; (c) have a history of experimenting with new research tools and processes; and (d) were willing to participate and easy to access.

Our research consisted of three core activities over two years: (1) observations and interviews; (2) participatory design of a prototype; and (3) in-situ interviews while using the prototype (fig. 38 right).

[34] **Klokmose2015**.
[35] Rädle et al., 'Codestrates: Literate Computing with Webstrates.'

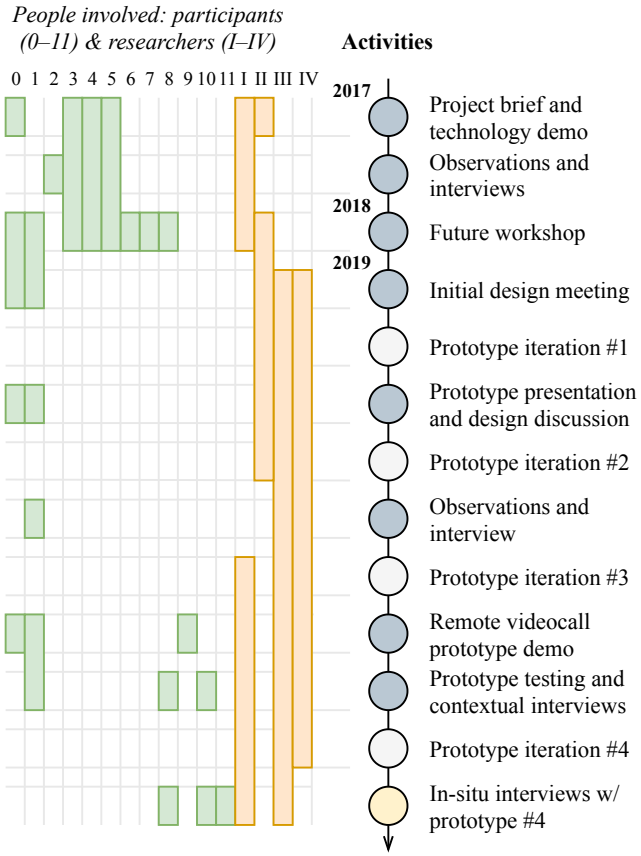


Figure 38. Overview of researcher (orange) and participant (green) engagement in the research process.

10.3.1 *Participants*

We collaborated with the Andersen Lab for Biomolecular Design[36] which is part of the Interdisciplinary Nanoscience Center[37] at Aarhus University. The nanoscience center is a consortium of fifteen departments and faculties across two universities, housing twenty-five research groups. At the time of the study, the lab consisted of one principle investigator (PI), eight postdoctoral researchers (one remote), and four PhD students. All members of the lab except one were involved in the study to different extents. The process overview in Figure 38 to the left shows the participant engagement in the different activities in the project.

10.3.2 *Observations and Interviews*

To gain a phenomenological understanding of the work practice of the participants, their use of computational tools, and the socio-technical context of the lab, we took a contextual inquiry-based approach, which totalled five whole days of observations at the lab and six formal in-context interviews. Before starting our

[36] https://bion.au.dk
[37] https://inano.au.dk

observations, we were introduced to the research group at their weekly meeting. We explained our overall goal and asked for consent to record their work. The participants were introduced to our general research agenda and we demoed previous prototypes of computational media that we had developed. We followed two participants for the entire duration of a particular experimental phase — each lasting between a day and a day and a half — and opportunistically observed and interviewed the other members of the lab when appropriate. During the experiments we tried to get a general understanding of the participants' work practice and asked them to walk us through the different steps of their experimental process, why they were doing it in the way they were, what problems they run into while doing their work, and what they would like to see changed. In particular, we asked them to report on the variety of computational tools they used and how they worked together. During the formal interviews, we used the critical incident technique[38] to ask the participants about recent or memorable technology breakdowns. The data collected in this phase included field notes, photos, videos of participants interacting with equipment and software, and audio recordings of interviews and workshops. The interviews were transcribed pure verbatim.

10.3.3 *Participatory Design of a Possible Future Prototype*

To help the participants reflect on how computational media might or might not be useful for their work practice, we wanted to establish a *possible future*,[39] that is, use prototype development to co-create a future vision of a computational laboratory notebook "empirically researchable in the present world".[40] The aim of this participatory design process was to encourage the participants to iteratively reflect on their current computational practice, crystallize ideas into prototypes, and give them hands-on experience with a possible future lab notebook.

The participatory design process started with a full-day workshop that focused on collectively imagining the future of digitally supported biomolecular nanoscience. We presented the empirical data we had collected and asked the participants to confirm and expand on our observations. Grounded in these accounts, the participants wrote detailed scenarios of one particular, common activity of their work practice. They were asked to bring up the different digital and analog materials they used, the breakdowns that occurred, and the workarounds they employed to deal with them. We picked two of these scenarios and ideated in two groups on how the scenario could be better supported digitally in the future. We chose one of these two ideas to design and develop into a prototype. The idea was chosen based on its level of significance to their research, and its technical feasibility to implement.

[38] John C. Flanagan (1954b). 'The Critical Incident Technique.' In: *Psychological bulletin* 51.4, p. 327. DOI: 10.1037/h0061470.

[39] Antti Salovaara, Antti Oulasvirta, and Giulio Jacucci (2017). 'Evaluation of Prototypes and the Problem of Possible Futures.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, pp. 2064–2077. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025658. URL: http://doi.acm.org/10.1145/3025453.3025658.

[40] Ibid.

We focused on the first part of their experimental workflow, which is the design phase of RNA structures. We conducted four meetings with researchers of the laboratory and a collaborator in the US to iteratively design and develop the prototype.

Using previously conducted fieldwork as a basis, we implemented the initial iteration of our prototype. It featured rudimentary functions like the displaying of a 3D representation of an RNA molecule or a drawing canvas for sketches. We used this first iteration as a way to communicate our ideas during the first meeting with two researchers of the lab. This meeting was used to get an initial reaction towards our idea of computational media and to clarify the exact steps of the workflow of one of the researchers.

After developing the structure of our prototype, during a second meeting, one of the researchers showcased how the tools and scripts are used in practice. This showcase was followed by a discussion on how and with what features the researcher could imagine this workflow using a computational medium. The third iteration of the prototype, which implemented various scripts and tools the researchers use, was discussed in a meeting with three researchers of the lab, two of whom were introduced to the prototype for the first time during that meeting. After a brief introduction to the prototype, all three researchers used it with one of the molecule designs they were working with at the time. Based on the feedback and ideas of the three participating researchers, the prototype was iterated another time. The fourth prototype was used as the basis for the in-situ interviews described below.

10.3.4 *In-situ Interviews While Using the Prototype*

We conducted in-situ semi-structured interviews with three participants while they used the prototype to design a new RNA structure. We chose these three participants because they, at the time of the study, had concrete work they needed to do which the prototype was specifically designed to support. The three independent sessions with the participants started with us giving them an introduction to the goals of the project and the process so far, a brief mention of the history of these types of software tools, and a few words on what HCI research is. We introduced the participants to the four principles that the prototype was built upon: shareability, malleability, distributability, and computability. We did this to allow them to reflect on these, and for them to have a vocabulary to talk about the prototype with. After the introduction, which lasted between 15 and 20 minutes, they received an in-depth demonstration of how the prototype worked.

We began the interviews by asking the participants to design an RNA structure using the prototype. The prototype was used as a *springboard*[41] into a discussion of their computational work, their thoughts on the four principles, and a general reflection on computational media in relation to their work. The interviews dealt with their general software problems, further explaining and discussions of the inner workings of the prototype, and even co-debugging their pre-existing scripts that had been imported into the notebook prototype. The interviews were guided

[41] Yrjö Engeström (2015). *Learning by Expanding*. Cambridge University Press.

by a number of questions we had prepared, e.g., "When was the last time you tweaked a script?" or "Do you have any shared documents or resources in the lab?". The interviews lasted 82 minutes, 57 minutes, and 90 minutes, respectively. All authors coded them independently (open coding), focusing on statements related to the four principles of computational media. We collected the codes and examples on a whiteboard and discussed how to arrange them in themes over three meetings.

10.4 FINDINGS

The findings are structured as follows: first, we give an overview of the nanoscience laboratory and explain the work and environment of the participants. Second, we describe three defining computational characteristics of biomolecular nanoscience that affect the participants. Lastly, we present the design of our computational labbook prototype and how it relates to the participants' work practice and computational challenges.

The findings are presented in an anachronistic order. The computational characteristics we describe came out of co-reflecting with the participants while using the prototype for a realistic work task. However, we present these insights before we describe the prototype's design, because we believe this makes it easier for the reader to understand and appreciate what challenges the prototype tries to address.

10.4.1 *Overview of the Lab*

Nanoscience is the study and application of materials and structures on the nanoscale, a transdisciplinary research category that can be found in fields such as chemistry, biology, physics, engineering, and material science. The research group under discussion here is one of the world's leading molecular biology labs specialising in the origami method: a way of creating molecular structures using DNA, RNA and proteins as building blocks which can assemble themselves into non-arbitrary shapes. This allows researchers to create "nanorobots"; structures that can perform pre-programmed tasks such as turn into a smiley, act as biosensors, or (hopefully in the future) deliver drugs to specific parts of the body.

The experimental process of this area of research can be roughly divided into two parts: the "dry" part, which includes the open-ended and creative design of new structures; and the "wet" part, which includes creating those structures in the lab and performing experiments on it to see how it behaves.

For the wet work, each of the researchers in this group has their own assigned workbench in the laboratory (fig. 39). The lab is crowded with scientific instruments they use for their experiments (e.g., vortex centrifuges, electroporators, non-confocal laser scanners). Because they work with genetic material, they have to be careful not to accidentally contaminate their samples. They wear latex gloves in the lab, and there are strict rules about "glove-on" and "glove-off" equipment. They also limit the number of things entering or leaving the lab as much

Figure 39. A lab bench in one of the main laboratories.

as possible to protect the environment, and anything brought from the outside is only allowed to be placed in specific taped off sections on the lab bench.

The dry work is predominately done at a desk in the office. Over the years, each researcher has accrued an ecosystem of computational tools they feel most comfortable with. Some of them have invested significant time designing their ideal software environment, with scripts to automate parts of their tasks, sophisticated Excel spreadsheets, and carefully chosen commercial and open-source web and desktop applications. Some of the researchers are provided with a laptop by the department, others simply bring their own device. Because of the restriction on moving things into and out of the lab, to use a computer in the lab means sacrificing that computer to that space, resulting in some of the researchers having one high-quality device for their office work and a second "disposable" laptop for the laboratory. All team members are also supplied with Apple iPads as part of the PI's efforts to digitise the lab and experiment with new ways of working.

There are two computational tools that are absolutely fundamental for the lab, the knowledge it creates, and the way they organise their work. The first is their ELN: a wiki-based collaborative software called Confluence. The notebook is the linchpin of the natural sciences, used to plan experiments, record research results, and document analyses. It plays an important part in ensuring scientific integrity and settling intellectual property disputes. The Confluence ELN — a digital remediation of the traditional paper notebook — started as a grassroots project in this group, but was later picked up by the university administration and is now used by most of the labs. Each member of the group has their own webpage on the ELN, which they use to plan and report their experiments, present results during

weekly meetings, hand over projects between researchers, and onboard new lab members. The second crucial computational tool is their repository of in-house developed Perl and Python scripts, which the researchers use to compute RNA structures, generate 3D visuals, and reformat outputs to be more readable.

These scripts were written by one particular postdoc who taught himself how to program, and they are used in good faith by other lab members who have fewer or no programming background. Without these scripts, the lab would not be able to do RNA origami research, since this is a frontier of science that is still being invented.

10.4.2 *Computational Characteristics*

We identified three characteristics of computational biomolecular nanoscience that shape how participants structure their work, their psycho-social working conditions, and scientific knowledge they produce related to their computational tools.

10.4.2.1 *Computational Culture*

Biomolecular nanoscientists have a distinct relationship with computers and computation compared to other, related scientific fields. Reflecting on the tools they use, P3 remarked:

> *Our lab is derived from molecular biology, so our methods are also from that field. We use "black box" machines a lot, where you put in your sample and just press start and you get your data. Physicists [on the other hand] like to have their own equipment they can tinker with.* (P3)

The sophistication of those "black box" machines — whether hardware or software — differs quite significantly depending on the particular research questions the scientist focuses on. If the area of study is well established, so is the computational support. But going beyond the frontier of an established type of research means stepping into a tabula rasa of software. P10, for example, was confronted with this jarring difference when switching from *analysing* RNA structures in their previous job to *designing* them when they joined the current lab:

> *We didn't execute a whole bunch of codes in our previous lab. I was more using pretty well established software with a nice-ish GUI [...] I was still doing RNA work, but I wasn't doing any design of RNA sequence or anything like that. So depending on the research you'll have either really well supported workflows ... or nothing ... or homebrew scripts.* (P10)

Switching research focus, then, also means a shift in the kind of computational competencies required of the researcher. For the ground-breaking RNA origami that this lab does, there are no well-established tools available. The lab members have to cobble together scripts in different languages, some of which are built in-house and others they found online. When asked if they could do this kind of research without those scripts, P11 answered:

> *No, no I couldn't do it. [...] I could do other kinds of science, but this is what is giving us like, a lot of chances to create new structures. [...] I see sometimes other people designing RNAs manually in some papers ... that's crazy. Sequence design manually? Do you know how crazy that is?* (P11)

The friction comes from the fact that the software skills necessary to understand, run, and debug those scripts are not (yet) part of the formal education leading up to this kind of research, nor is it considered part of the professional culture to pick them up. Biomolecular nanoscientists with the computational literacy to produce or modify scripts are "pretty rare" (P10) and learning how to program is considered "a whole other career" (P11) rather than an integral part of their job. As a result, the development and maintenance of the computational tools vital to the participants' research rests on the very precarious foundation of a once-in-a-blue-moon computationally literate graduate student, postdoc, or research assistant.

In summary, the computational culture of biomolecular nanoscience is one where there is a dependency on computational tools to do the work, but not a tradition of training scientists in the development, deployment, and maintenance of them. This is fine when doing established research using GUI software, but becomes very restricting when working on a frontier of science where the tools have to be invented. There is an incongruous need for the flexibility of scripts that allows researchers to invent new methods, but the usability and convenience of applications. A computational medium would have to address this gap between flexibility and usability.

10.4.2.2 *Computational Environment*

The biggest challenge for the nanoscientists was not a lack of coding skills, but having to manage the computational environments in which to execute code. Because there are few computational tools that support their work, the participants use any software — whether an open source system published on GitHub or a script sent by a colleague from a different lab — regardless of language, dependencies, or quality of documentation. Installing, maintaining, and debugging the various environments that these tools rely on, however, has proven to be categorically prohibitive. Issues include terminal commands that change across operating systems, having to set up virtual environments to run scripts, fixing corrupted installations of language interpreters, etc.

The consequences of these challenges are that they work with partial outputs, accept bottlenecks, and even abandon certain research approaches. For example, P8 explains how they had to give up on a popular tool because they could not get it to work in Windows.

> *I was trying to get this software that everyone was using [...] and I couldn't get it to work [...] Probably because it's for like Linux and my Windows just didn't ... so I had to either do a virtual machine, get Linux, and do it from there ... but then I just gave up.* (P8)

P11 tried to run a script given to them by a collaborator but could not get it to work[42]. In the end, P11 simply sent the data and asked the collaborated to run the script on it and email back the result.

> *We are trying to build this new structure, a triangle that is a bit weird. So what I do is, I try to design the pattern that I think is going to fold the way we expect and then I send it to him and he runs it, and he's like "no".* (P11)

In summary, using computational tools requires an understanding of the entire software stack, but this kind of information is often assumed in documentation and few online tutorials focus on such obscure dependency management. A computational medium should break the tight coupling between hardware and computational environments, and instead merge the computational environment with the code. This would allow the code/environment artefact to be self-contained and easily shared regardless of the underlying platform.

10.4.2.3 *Computational Disempowerment*

Paradoxically, the very tools that provide the scientists with the ability to push the boundaries of RNA origami also created a psycho-social working environment where they frequently felt disempowered. Participants were vocal about how their tools made them feel helpless, but mostly placed the blame on their own perceived lack of skills rather than the design of the technology.

> *So now I'm following this tutorial on how to use this other script and it doesn't work and that's it. I have no idea what to do. I don't understand how it works, at all.* (P11)

When a script reports an error, it is in the form of error codes or snippets of a foreign programming language. Errors in the molecular structure that make it impossible to be created are expressed as software errors and have to be somehow deciphered back into the realm of molecular biology.

> *This is one of my points of frustration. Because I have no idea what that [highlights terminal error] means [...] I have tried to to look at the script. And that's about as far as I got. Because this means nothing to me. So if I go to line 2856 ... 2856 is ... print oped spool, okay yeah that's what's there [laughs].* (P10)

In fortunate situations, a more capable peer can assist, as when P11 explained how they would send data to a remote collaborator to run a script that would not work on their own computer. However, this is a fragile workaround as P11's work became dependent on the collaborator's time and goodwill.

Sometimes it is obvious that there are more efficient ways to go about a problem, but time and skill prohibit doing something about it.

[42] We helped fix the issue during the interview: the script would overwrite the original source file instead of creating a new file.

> *I think this is one of those things where it's definitely not the most efficient way of doing this, but it'd take me longer to find a more efficient way.* (P10)

P8 explains how one of the scripts requires not having duplicates of the same sequences, but is a bit uncertain whether it is a biological or software limitation.

> *RNA wise it makes sense. But for some of these it doesn't ... I kind of like having duplicates.* P8

Even though P8 would like to be able to change the script they don't, instead they "hack the structure a bit" (P8).

The new computational possibilities are a blessing and a curse. They enable scientific breakthroughs that were previously inconceivable, but also create a dependence on machinery that the scientists are not trained to operate. When breakdowns happen — and they happen often — the scientists are largely left to their own fate, to much frustration.

10.4.3 *The Computational Labbook Prototype*

We produced a prototype through which to co-reflect with participants on their current and imagined future computational tools and activities. The findings presented above largely came from the co-creation of the prototype and its subsequent role as an artefact to talk about their current challenges and possible futures. In this section, we will describe the particular task that this prototype was developed for, how it implements the computational media principles suggested by previous work, and how its design addresses some of the three challenges faced by the scientists mentioned in the previous section.

10.4.3.1 *Supporting the Designing of Macromolecules*

Our prototype of a computational labbook[43] is designed to support the design of RNA nanostructures, also referred to by the scientists as the "dry" part of the experimental process. This phase typically consists of some variation of these steps:

1. Producing a 2D textual representation of the target macromolecule, often by adapting an existing structure (from previous work or another lab member) with a molecule that folds in a desired way.
2. Checking whether the structure is physically capable of folding in the desired way by creating a 3D representation of it and looking for flaws.
3. Computing possible genetic sequences that might fold into the desired macromolecular structure and ranking them by viability.
4. Turning the 2D textual representation into a string-based representation that can then be send to a commercial company when ordering.

[43] See the accompanying video for an overview of the prototype.

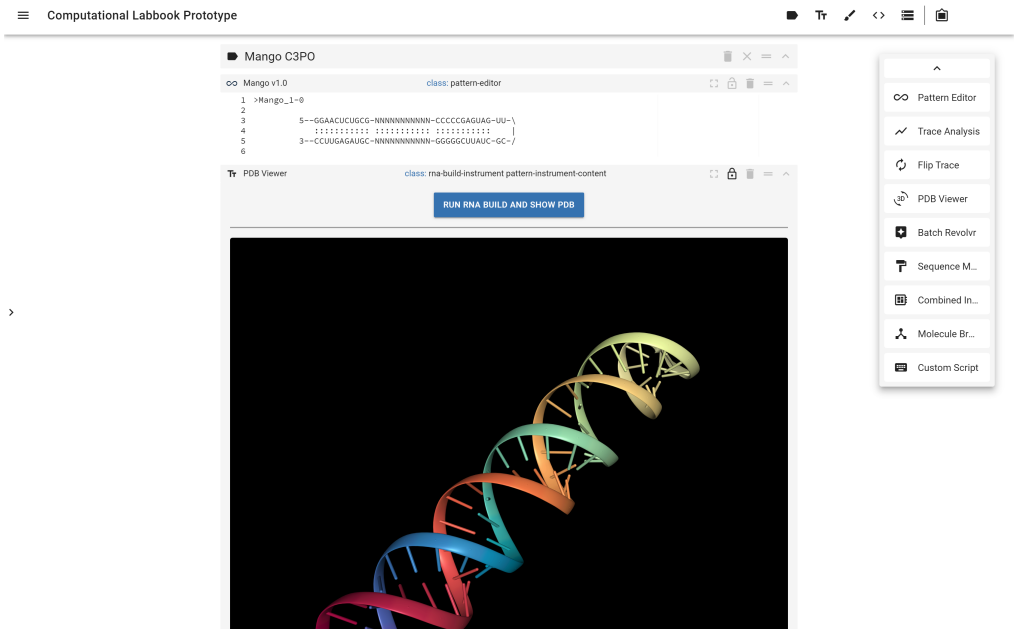


Figure 40. Screenshot of the computational labbook. The center shows the sections and paragraphs of the document; on the right side the *Instrument Panel* allows users to drag instruments into the document.

Step two and three are largely interchangeable, and at least one researcher does not create a 3D representation at all. The "dry" part is then followed by the "wet" part in which researchers assemble, analyse, and validate their work in the lab space.

10.4.3.2 *Redesigning the Task as a Computational Notebook*

Besides providing functionality typically associated with ELNs, such as text editing and image uploading in a document, the computational labbook prototype lets the user add various computational instruments to the document by dragging and dropping from an instrument panel (fig. 40).

The prototype addresses the steps of the dry phase in the following ways: To design a new structure (step 1), the scientist can create a new document and drag in an ASCII-based *Pattern Editor* instrument for the textual representation of the RNA structure. To preview the structure (step 2), a *PDB*[44] *Viewer* instrument can be added, which generates an interactive 3D visualization of the structure in the document. Now, the *Batch Revolvr* instrument can be added to compute multiple candidates of molecules (step 3) for a specific pattern scaffold by — broadly speaking — repetitively filling the scaffold and computing its energy properties. This is necessary as it is not feasible for researchers to try out every possible combination of nucleobases[45] in their scaffolds which can contain hundreds of nucleobases. The resulting candidates can then be used for further analysis in other external tools. As the computation of candidates is computationally heavy,

[44] The Protein Data Bank file format.
[45] Put simply, nucleobases are the four fundamental building blocks of DNA/RNA molecules.

it is offloaded to a server which — once finished — sends the results back to the document.

The ability to offload heavy computation to a fast central server was (unsurprisingly) something that was well received by the participants.

> *It works pretty well! [laughs] This is so fast. This is really good.* (P8)

Other instruments can be added to perform analyses on the structure, to perform various sequence operations (step 4) such as converting from RNA to DNA or reversing a sequence, or to give access to a shared repository of molecule designs.

> *Having the repository is nice because it's just click and drag and not a text document, stuff like that. But having it shared is also nice because you can make sure people are using the same motifs, like this mango one [a particular structure] has been through like five iterations.* (P10)

Multiple designs can be made in a document, and the computational instruments can be interleaved with written text, images, and even hand-drawn sketches if the notebook is opened on a tablet with a stylus. The unification of the different tools and scripts in one document was particularly highlighted by the participants.

> *[whispers] Oh this is really convenient . . . usually I would have to like go in . . . make the modification, save the file, run the script on the terminal, wait for it to generate the file, then click that.* (P8)

> *It would be nice to have the whole . . . everything from trace analysis to Revolvr all on there. I think it would be nice to do . . . because it's all amalgamated into one location. And it's a well documented workflow rather than having . . . [clicks and shows folders of files] this.* (P11)

Using and discussing the prototype also spawned new ideas, e.g., the ability to chain the output of one script to the execution of another.

> *This is pretty fast! Except you don't get the summary, I would love to get these results but run another script on it.* (P8)

10.4.3.3 *Adherence to the Principles of Computational Media*

The prototype was developed to embody the four principles of computational media: shareability, distributability, malleability, and computability. We realised these principles partly by leveraging existing features of Webstrates and Codestrates and extending them with new ones. Table 7 gives an overview over how the prototype practically implements these principles through specific functionality.

The principles are realised in the following manner: Notebooks are *shareable* as they are accessed through their URLs, they can be copied, and they support real-time collaborative editing and interaction. Nanostructures can be shared between notebooks through a repository. Also, functionality can be shared between

notebooks using its built-in packaging system from Codestrates. While the participants didn't collaborate closely on a day-to-day basis, they reflected on how the shareability could help in knowledge sharing.

> *I think this is just so much nicer, just from an organisational point of view. Like if he's working on these designs, instead of just sending me the text files and then I have to organise that in my own computer and yada yada, I can just look at his labbook and see the output, and see what he's done.* (P10)

The prototype is web-based and can be accessed from any device with a web browser, and as a result it is *distributable* across a variety of devices (e.g., desktop computers, laptops, tablet, and phones) and operating systems (e.g., Windows, macOS, Android, and iOS). It supports distribution of interaction where, for example, the PDB Viewer can be shown on a tablet while the structure is edited on a laptop. Also, it supports distribution of computation. While the benefits of distributing computation was obvious to the participants, it was harder for them to imagine how to leverage distribution of interaction across devices in their day-to-day work.

The prototype supports *computability* as computations can be executed directly in the documents of the labbook without any setup, and it supports the execution of multiple programming languages. Compared to a conventional computational notebook, the prototype does not have code front-and-center. The code for the computational instruments *can* be accessed and modified from within the notebook, including the Perl and Python scripts developed in the lab. The built-in versioning support in Webstrates enables to revert to a working state if the notebook breaks. The notebook includes an instrument for creating custom scripts in Python, two dialects of Perl, Ruby, or Node.js JavaScript. The execution of the custom scripts is offloaded to a server. Hereby, the notebook supports *malleability*.

10.4.3.4 *Technical Foundation*

The prototype is realized using Webstrates[46] and its authoring environment, Codestrates.[47] Webstrates[48] is a platform for creating computational media that follows a document-centric approach to software. Documents in Webstrates are identified by their URL and can also be shared using it. Every document is self-contained and includes the source code of its own implementation. In Webstrates, changes to the document are made persistent on a server and synchronized to all other clients with the same document open, this includes changes to embedded scripts. The user interface of the labbook prototype is an adaptation of the Codestrates user interface following the same document-centric structure of sections and paragraphs, where paragraphs can be textual, data, executable code, or user interface elements. Functionality can be dynamically added and removed from

[46] **Klokmose2015**.
[47] Rädle et al., 'Codestrates: Literate Computing with Webstrates.'
[48] **Klokmose2015**.

Distributability	– Documents accessible on any device with browser – Distribution of notebook paragraphs to other devices – Off-loading script execution to server
Shareability	– Real-time synchronisation of document changes – Package system for sharing tools – Repository of molecule structures – Easy sharing through URLs – Copying and branching of documents/templates
Malleability	– All code is accessible and changeable from within – Version control as safeguard – Custom scripts
Computability	– Scripts, data, and output in the same document – No setup of computational environment required – Execute code in multiple programming languages

Table 7. Overview of how the computational labbook prototype realises the four principles of computational media.

a from a document using Codestrates' package manager.[49] Each computational instrument is implemented as a package, which means that it is possible to configure a notebook with a specific selection of instruments, and also extend it with new ones.

The Codestrates platform supports the execution of JavaScript code directly in the document. The scripts of the nanoscience lab, however, are mainly implemented in Perl and Python. To be able to execute these scripts as well, we extended the Codestrates platform to make use of an experimental distributed computing platform for Webstrates. The latter uses Docker[50] containers on a dedicated server to run scripts of arbitrary programming languages in a sandboxed environment.

10.5 DISCUSSION

The design of the tools used by researchers play an important epistemological role in shaping what knowledge can be created. In his book *Changing Minds*,[51] diSessa explains how the invention of modern algebra at the close of the 16th century suddenly made it possible for secondary school level students to grapple with mathematical proofs that were previously only accessible to university-educated mathematicians. A parallel can be seen in the tools for the origami method in biomolecular nanoscience: modeling nanostructures was initially done by hand, but this limited the research to only small, manageable sequences. For

[49] Marcel Borowski, Roman Rädle, and Clemens N. Klokmose (2018). 'Codestrate Packages: An Alternative to "One-Size-Fits-All" Software.' In: *CHI EA '18 Proceedings of the 2018 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. DOI: 10.1145/3170427.3188563.

[50] Docker (2019). URL: https://docker.com.

[51] diSessa, *Changing Minds: Computers, Learning, and Literacy*.

DNA origami, a group of researchers released a software packaged called caDNAno that provided a graphical user interface for designing 3D nanostructures.[52] All of a sudden, this branch of research became more accessible and more popular, to the point that "undergrad students can now even build these types of structures" (P3). For RNA origami, which the researchers in this study focus on, the tools are in-house developed scripts that compute suggestions of RNA structures. Because the script was designed to process the inputs sequentially, when it gets stuck in an infinite loop on a particular structure, all possible nanostructures that might have been "discovered" afterwards are rendered scientifically impossible.

Our findings highlight the dichotomy between computational tools designed as scripts and applications. Software as a script provides the nanoscientists with great expressiveness and flexibility to do research on the frontier of their discipline, but becomes unapproachable because of the competences required to create, execute, and maintain it. Software as an application is more accessible because it requires less computational literacy, but they provide only turn-key operations that calcify a particular scientific praxis and the types of research that can be done. We could teach all scientists to also be software engineers so they could make their own tools, or we could provide all labs with professional programmers to create user-friendly applications. These are both commendable, albeit unrealistic paths. A scaleable solution is to develop a software model that straddles the divide of the dichotomy. This means questioning the application paradigm of the last 50 years and reinvigorate the efforts in creating computational media.

What design should computational media have? With our prototype, we applied four design principles derived from Klokmose et al.[53] and Rädle et al.[54] as a starting point: distributability, shareability, malleability, and computability. The rest of our discussion is structured around these principles, their relevance for the context of biomolecular nanoscience, and their limitations.

10.5.1 *Distributability*

Based on our observations and interviews, we can extend the concept of distributability for computational media by dividing it into three: distribution of documents, functionality, and computation. The distribution of documents is important because the participants need easy access across devices (laptop, desktop, tablet, phone) and spaces (office, home, laboratory, conferences). This need is already well-integrated into th mental model of the researchers, and facilitated by file-sharing platforms and the Confluence system. The distribution of functionality refers to the orchestration of operations across devices. Currently, operations are bundled in applications which means that each device needs to install the same application for functionality to be distributed (i.e., to access spreadsheets stored in the cloud, *every device* needs to install a spreadsheet application). How-

[52] Shawn M Douglas et al. (2009). 'Rapid prototyping of 3D DNA-origami shapes with caDNAno.' In: *Nucleic acids research* 37.15, pp. 5001–5006. DOI: 10.1093/nar/gkp436.
[53] **Klokmose2015**.
[54] Rädle et al., 'Codestrates: Literate Computing with Webstrates.'

ever, the participants did not engage in any multi-device activities (e.g., using the PDB Viewer on a tablet while using the Pattern Editor on a desktop computer) despite being provided with extra tablets by the lab. Previous research shows that — even if it is beneficial to use multiple devices to perform a task — users tend to use only one or at most two devices at a time.[55] Distribution of computation was something that made an immediate positive impact and was wholeheartedly embraced by the participants. Being able to offload heavy computations to a remote server was a crucial improvement over their current workflow, and is also a core focal point of many e-Science efforts. This, however, also necessitates server infrastructure that needs to be set up, be secure, should be scalable to a large number of users, and needs to be maintained.

10.5.2 *Shareability*

The participants work independently most of the time, so they did not see the need for real-time collaboration on a day-to-day basis. Instead, what the participants needed to be shareable was self-contained, computational environments. The difficulties they experienced trying to run scripts across devices make sharing their tools and results a huge hassle, either when when calling in the help of a more capable peer to debug an error, collaborating with a remote colleague, or networking at a conference. The problem of the tight coupling between computational environments and the hardware it runs on was similarly highlighted by Guo and Engler.[56] They built the CDE system, which combines code, data, and environemnt into software packages that could be transferred seamlessly between Linux machines. Our prototype facilitates shareability by storing and running documents/environments on a server that are accessed through URLs, which is a conceptual switch from the traditional pass-by-value of sharing to a pass-by-reference model. Beyond this, however, the prototype does not implement any functionality that supports collaboration. Research into groupware (see[57] for a historical overview) shows that when multiple people can access a document, questions such as "Who did that?" or "What happened since I was here last?" become pertinent.[58] We see additional challenges when the document is computational, with questions such as "Is something computing?" or "What computation is this a result of?". These need to be addressed in future designs, especially for

[55] Thomas Plank et al. (2017). 'Is Two Enough?! Studying Benefits, Barriers, and Biases of Multi-Tablet Use for Collaborative Visualization.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. DOI: 10.1145/3025453.3025537.

[56] Philip J. Guo and Dawson Engler (2011). 'CDE: Using System Call Interposition to Automatically Create Portable Software Packages.' In: *USENIXATC'11 Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, p. 21. URL: https://dl.acm.org/citation.cfm?id=2002202.

[57] Jonathan Grudin (1995). 'Groupware and Social Dynamics: Eight Challenges for Developers.' In: *Readings in Human–Computer Interaction*. Elsevier, pp. 762–774. DOI: 10.1016/B978-0-08-051574-8.50079-0.

[58] Carl Gutwin and Saul Greenberg (2002). 'A Descriptive Framework of Workspace Awareness for Real-Time Groupware.' In: *Computer Supported Cooperative Work (CSCW)* 11.3-4, pp. 411–446. DOI: 10.1023/A:1021271517844.

scientific contexts where provenance of data and computation are important to attribute authorship and settle intellectual property disputes.

10.5.3 *Malleability*

The kind of software malleability we observed was not a constant low-level tinkering with the code. Rather, it was similar to the malleability of a house, where adjustments can be made when needed, but which are often outsourced to skilled workers often such as carpenters and plumbers. As such, the concept of malleability needs to take a more collective view of software transformation, especially when the computational media targets users who are not trained programmers but rely on more capable peers. We implemented this by leveraging the package system of the Codestrates platform, which makes the medium extensible through packages authored by others. This allows users with limited technical skills to mold their environment, while retaining easy access to the code should it be needed.

There is also the question of whether malleability is actually desirable, especially in the context of laboratory notebooks. While this did not come up explicitly in our process, Oleksik et al.[59] document how the ease with which ELNs are edited clashes with the "requirements for persistence and consistency of scientific records" and makes collaborating with others more difficult because document structures are no longer standardised. They suggest such systems could implement "fixity" to address the first concern: features that allow users to freeze certain states of a document so it can be authentically replicated. To facilitate collaboration, they recommend adding functionality that can deal with multi-structured data, such as text mining, natural language processing, and scheme integration.

10.5.4 *Computability*

Because the participants operate in a space where there is no established software ecosystem, they opportunistically use scripts from different sources, written in multiple languages. Therefore, supporting computability in a computational medium should not just be about the ability to execute code, but should also allow the agnostic mixing of multiple programming languages in the same space. Our prototype distributed code execution to containers on a remote server, which makes it in principle possible to run any code. However, this simply moves dependency management to a device out of the control of the user, which does not scale very well without requiring the help of experts to maintain those execution environments. With multiple, heterogeneous components also comes the need for piping the output of one into another — something participants requested.

[59] Oleksik, Milic-Frayling, and Jones, 'Study of Electronic Lab Notebook Design and Practices That Emerged in a Collaborative Scientific Environment.'

While our current prototype does not support this, we have developed reactive data-flow pipeline mechanisms using Codestrates elsewhere.[60]

Ideally, the day-to-day access to a medium that supports computability will improve the computational literacy of its users over time. There is evidence that this happened in the early days of the web through sites such as Myspace and Neopets. These social media helped users pick up HTML and CSS skills simply because the code was accessible and could be tinkered with.[61] Whether such learning-through-exposure extends to imperative languages such as JavaScript, Python, or Perl is unknown to us. What we do know is that creating the type of interactive components that we have developed for the prototype requires being able to read and write asynchronous event-based code, which is difficult to pick up by simply stumbling into an existing codebase.

10.6 CONCLUSION

The tools used in research shape the science and the scientist, guiding what knowledge can be created and which competences are required to create it. In this paper, we explored the scripts and applications used in biomolecular nanoscience, and the concept of *computational media* as an alternative software model that sits in between them. We discussed how the *computational culture* of biomolecular nanoscience creates a dependency on digital tools, but does not have a tradition of training the scientists in developing, deploying, and maintaining them. We showed how, contrary to what most policy around the digitalisation of science focuses on, the nanoscientists do not need help with learning how to code, but with managing the *computational environments* required to execute that code. As a result, the tools that make their research possible also are a source of feeling *computational disempowerment* on a daily basis.

Computational media — systems where users can author and execute code in the same perceptual space as other multimedia content — could provide biomolecular nanoscientists with the flexibility of scripts together with the accessibility of applications. During a two-year participatory design process with the Andersen Lab at the Aarhus University iNANO Center, we built a computational labbook prototype to explore this potential. We used the prototype to critically evaluated four design principles for computational media suggested by previous research: distributability, shareability, malleability, and computability. We find that distributing functionality across devices is not particularly relevant, but distributing (heavy) computation is crucial; that being able to share self-contained execution environments could play a small but critical role when collaborating and debugging code with more capable peers; that malleability is a collective rather than individual need; and that computability should support the execution and coupling of multiple programming languages.

[60] Sriram Karthik Badam et al. (2018). 'Vistrates: A Component Model for Ubiquitous Analytics.' In: *IEEE Transactions on Visualization and Computer Graphics*. ISSN: 10772626. DOI: 10.1109/TVCG.2018.2865144. URL: https://karthikbadam.github.io/assets/data/vistrates.pdf.

[61] Dan Perkel (2008). 'Copy and Paste Literacy: Literacy practices in the production of a MySpace profile.' English. In: *Informal Learning and Digital Media*. Ed. by Kirsten Drotner et al. Cambridge Scholars Press. Chap. 10, pp. 203–224.

For the past forty years, applications have been the dominant model for software. But the emergence and popularity of code-centric computational notebooks (e.g., Jupyter) and no-code computational documents (e.g., Notion) are exciting hints of a changing paradigm. Without a dominant design in place, what these computational media will look like is still an open challenge. Although our study is rooted in a very particular context, we hope it inspires a larger discussion around the paradigm shifting potential of this type of software.

11.

CONCLUSION

Denmark is one of the few countries in the world which regulates its labour market not through legislation, but through tripartite discussions and collective agreements between employer organisations, labour representatives, and the government. At its very heart, the "Danish model" is about the *negotiatibility* of labour conditions such a wages, pensions, hours, and the quality of working environments. It requires that all stakeholders have strong, representative organisations with a seat at the table that can bargain for their members' interests.

Denmark is also one of the most digitised countries in the world, following thirty years of successful national strategies. Reports such as the European Commission's Digital Economy and Society Index[1] and the UN's e-Government Survey[2] consistently place Denmark in the top three digital societies, with high levels of broadband connectivity, widespread integration of technology across industries, and trusted digital public services.

The current state of informational capitalism has placed these two characteristics of Denmark's labour market – the high-level of digitisation and the negotiation-based regulation model – on a collision course. On the one hand, digitisation means that work activities are increasingly mediated by software, and the design of that software affects the labour conditions of Denmark's workers. For example, in Chapter 8 I described how the perceived labour power of Danish knowledge workers is strongly connected to specific applications, but that the lack of interoperability between applications undermines that power when collaborating with others and affects their psychosomatic health. On the other hand, Denmark's regulation of working conditions through tripartite negotiations and collective agreements is incompatible with the software used by its labour force. In Chapter 5, I traced the historical construction of the application model of software and showed how its negotiability was eroded by the process of commodification. It transformed from a cooperatively designed artefact in the 1950s, to a commissioned package in the 1960s, to a mass-market product in the 1970s and 1980s. Throughout this process, the developers of the software have increasingly appropriated control over its design: source code was hidden away, data formats made proprietary, interoperability abandoned, and interfaces copyrighted. On top of the way the application model centralises control with its developers, in Chapter

[1] European Commission, *Digital Economy and Society Index (DESI) 2020 Denmark*.

[2] United Nations Department of Economic and Social Affairs (2020). *United Nations E-government Survey: Digital government in the decade of action for sustainable development*. United Nations.

7 I reported on the extreme homogeneity of applications used by Danish knowledge workers. This further concentrates the power over Denmark's digital working conditions in the hands of a small group of US American corporations with no moral commitment or legal responsibility to participate in tripartite negotiations.

Alternative software models are possible that, at least technologically, open up the possibility for distributed control over the design of software. In Chapter 9 I presented Codestrates, a computational medium that removes the distinction between the use and development of an application and allows users to collaboratively and continuously change its design using a literate computing approach. In Chapter 10 I described how we further developed this software model together with a group of biomolecular nanoscientists, showing first how the lack of negotiability of applications forces them to revert back to command-line interfaces, and then how the Codestrates-based platform provides them with the usability of applications combined with the flexibility of scripts that they require for their work.

Part I of this dissertation asked whether the power and control of large multinational technology corporations could be redistributed by making software negotiable by design. While this line of research was successful academically, it did not affect a meaningful, lasting change in the digital labour conditions of knowledge workers.

HCI research – including this dissertation – generally sets itself apart from other technology-centric research disciplines by focusing on the user as the main object and beneficiary of study, and on design as the vehicle for delivering those benefits. Its overall theory of change is that establishing informational harms and developing research prototypes that address those harms will trickle down into computational products and subsequently solve the identified problems. Following this tradition, I have successfully documented the undesirable characteristics of the application model of software, and demonstrated the technological feasibility of *negotiable software*. This knowledge might have affected a change at a time when workplace software was primarily developed in-house, meaning researchers such as myself had a direct line of communication with the organisation that would ultimately implement the system. However, because most applications used by workers are produced by inaccessible developer kings, the unavoidable conclusion is that such user-centric, participatory design approaches are essentially assemblies of the powerless. Without a way to inform or enforce changes to existing software products, the knowledge generated by these research projects will make a difference only by accident.

Part II

Negotiation Software

12.

INTRODUCTION

There is no easy path towards change in complex social systems, and believing technology can unilaterally solve problems equates to unrealistic (and according to some, destructive[1]) solutionism. However, that does not mean that digital technologies cannot be *part* of the solution.

Part II of this dissertation describes the use of *negotiation software*, i.e., software that supports negotiating activities with other actors in complex social and political systems. The software that I will target are consent manangement platforms (i.e., consent pop-ups) in the context of European data protection regulation.

12.1 SOFTWARE: CONSENT MANAGEMENT PLATFORMS

Under informational capitalism, the primary source of surplus value is the knowledge generated from the processing of information.[2] Before information and knowledge, however, there are data: "material produced by abstracting the world into categories, measures and other representational forms".[3] The fundamental role of data in informational economies has given rise to two kinds of processes: on the one hand, more and more aspects of human life are *datafied*; and on the other hand, human life is transformed to render it more *datafiable*.[4] This dual move gives lie to the popular metaphor that "data is the new oil" as if it was a raw, naturally occurring resource that only needs to be extracted. Rather, Brown and Marsden propose, we should think of data as silk, and humans as the worm whose activities help produce the informational tapestries that sustain the empires of a few merchants, infamously Google, Amazon, and Facebook.[5]

The internet and the web are two technologies where the process of datafication has considerably transformed their original decentralised and (largely) anonymous design into platformed data-production machines. The business model of almost every contemporary online service relies on the creation and collection

[1] Evgeny Morozov (2013). *To save everything, click here: The folly of technological solutionism*. Public Affairs.

[2] Castells, *The Rise of the Network Society*, p. 17.

[3] Rob Kitchin (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage.

[4] Ulises A Mejias and Nick Couldry (2019). 'Datafication.' In: *Internet Policy Review* 8.4.

[5] Ian Brown and Christopher T Marsden (2013). *Regulating code: Good governance and better regulation in the information age*. MIT Press.

of behaviour and identity data, piped through intermediary brokers into real-time bidding infrastructures. Counter-technologies have been created that try and control the collection and flow of that data, for example browser extensions such as Privacy Badger and uBlock and HTTP protocols such as Do Not Track and Platform for Privacy Platform.

One particular sociolegal technology that has become the nexus of these power struggles are *Consent Management Platforms* (CMPs). These JavaScript libraries serve two purposes. On the front-end, they serve privacy notices for the website visitor to interact with and provide their consent through. On the back-end, they let the website owner integrate existing advertising services (e.g., Google AdSense), manage which tracking technology is added to the page at what point in time (before or after a visitor's interaction); and store those consent signals in a way that allows them to be audited.

12.2 CONTEXT: EUROPEAN DIGITAL RIGHTS AND RESPONSIBILITIES

The current state of data protection law in the European Union is the culmination of fifty years of legislative efforts, evolving from regional laws, to national laws, to EU laws, to fundamental rights in the EU Charter of Fundamental Rights.[6] The normative commitment of the EU to see data protection as a fundamental right, coupled with its legacy of civilian power,[7] has created a situation where control over data processing software is managed through legally protected rights and legally enforced responsibilities. Under this regulatory regime, there are four main parties who collectively shape this slice of digital life.

- The *data subject* or *user*, which refers to the natural person who can be identified through the data or is using an electronic device.
- The *data controller*, which refers to the entity that determines the purposes and means of the processing of personal data.
- The *data processor*, which refers to the entity that processes personal data on behalf of the controller. Depending on how much influence they have over determining the purposes and means of processing, they might be "upgraded" to a join-controller.
- The *supervisory authority*, which refers to the independent body responsible for monitoring and enforcing the regulation

Each entity has their own set of rights and responsibilities which they can exercise to co-regulate how data is collected and processed, including consent management platforms. These are the parties I will negotiate with.

[6] Gabriela Zanfir-Fortuna (2020). *Why data protection law is uniquely equipped to let us fight a pandemic with personal data*. URL: https://pdpecho.com/2020/04/06/why-data-protection-law-is-uniquely-equipped-to-let-us-fight-a-pandemic-with-personal-data/ (visited on 07/08/2020).

[7] Ian Manners (2002). 'Normative power Europe: a contradiction in terms?' In: *JCMS: Journal of common market studies* 40.2, pp. 235–258.

13.

A BRIEF HISTORY OF WEB TRACKING AND CONSENT POP-UPS

Tracking people on the world wide web and the consent pop-ups that purport to give people some control over being tracked, are two later additions to the design of network technologies. The purpose of this chapter is to trace the different technological and legislative decisions that were made which resulted in "Notice and Consent" being the dominant design of global, digital interconnection under informational capitalism.

13.1 THE INVENTION OF TRACKING

The design of the HTTP protocol – the back-bone of the web – did not originally include any way for a server to keep track of a user across different web pages. The reason for this design choice was to reduce interdependence and thus complexity, while increasing resilience. The successful handling of a single HTTP request-response cycle does not depend on any previous connection between a server and a client. It avoids the web server having to store any data between concurrent connections, or needing some kind of memory clean-up mechanisms. This *stateless* design makes it easier for the technology to scale, but it also makes it impossible to track a specific person across different web pages.

Both commercial companies and regulators were unsatisfied with this design. In 1994, the Bavarian State Government in Germany wanted to regulate porn on the internet. At the time, Bavarian citizens were serviced by CompuServe, the first US American online service provider, which did in fact host some porn on its servers. Because of the stateless design of the web that made it impossible to know who someone was and where they were coming from, the only way CompuServe could comply with the Bavarian regulators and avoid being punished was to block access to the porn for all worldwide customers of their network. Because of the design of the technology, the geographical scope of a single nation's regulatory action was global, something which drew considerable critique from international news media and digital rights organisations. About a year later, it implemented the technology to filter access to content on a country-by-country basis, making one of the first steps in the web's history of tracing digital connections to physical people.

Also in 1994, the browser company Netscape wanted to make it possible for people to place items in a digital shopping basket and for that information to

be remembered when they went to a new page, i.e., the check-out. The stateless design of the web made such actions impossible, which was seen as a considerable barrier to making e-commerce (and through it, the entire web) financially viable. Montulli, a founding programmer at the company, was given the task to come up with a technological solution. He ended up using the header of an HTTP request-response to send data from the server to the client's device, which the Netscape browser would then read and save: the Set-Cookie Response Header. Netscape shipped this implementation for the first time in September 1994 as part of their Mosaic browser.

This implementation managed to spread far beyond the Netscape browser through the www-talk mailing list of the World Wide Web Consortium (W3C). In April 1995, Brian Behlendorf, a programmer at Hot-Wired.com, noted the general interest of commercial companies in people's "clickstreams" through websites, and proposed a Session-ID HTTP header to store these. Montulli replied saying they had been "doing some work along these lines",[1] and included his proposal:

```
Syntax of the Set-Cookie HTTP Response Header:
Set-Cookie: NAME=OPAQUE\_STRING \
    [; expires= ] \
    [; path=] \
    [; domain=] \
    [; secure]

Syntax of the Cookie HTTP Request Header:
Cookie: NAME=OPAQUE\_STRING *[; NAME=OPAQUE\_STRING]
```

The proposal fell on fertile ground. In December 1995 a subgroup of the W3C HTTP Working Group was created around this idea of "state management". It used Montulli's proposal as a starting point and published the specification of cookies as a memory mechanism under the name HTTP State Management Mechanism in February 1997. Two important and frictious design decision were made. First, cookies were rejected by default and required the user to opt-in. Second, it would only allow cookies to be placed if they came from the same domain the user was currently visiting. If the webpage used third-party sources for certain elements on the page (e.g., images), the HTTP response sent along with that content would not be allowed to set a cookie on the user's device.

When this specification was published, two organisations joined a conversation that normally did not receive a lot of outside attention. DoubleClick, the largest and already contentious internet advertising agency, argued that blocking third-parties like themselves from placing cookies and tracking users across websites would have "huge ramifications" for the economic viability of small websites that relied on selling advertising banner space to fund their operations. The second organisation, the Electronic Privacy Information Center (EPIC), occupied the other side of the spectrum and came out in support of the specification. It concluded its letter of support by saying that transparency – "the ability of users to see and exercise control over the disclosure of personally identifiable information" – was the most important principle to protect when designing these kind of tracking

[1] Lou Montulli (Apr. 18, 1995). *Re: Session tracking*. www-talk electronic mailing list message. URL: https://lists.w3.org/Archives/Public/www-talk/1995MarApr/0462.html.

technologies: third-party cookies would be difficult for the user to keep track of and understand, making it untransparent and undesirable.

The disagreements around the technical implementation stalled the acceptance of the proposal, which continued to be iterated over until at least 2000. In the end, Netscape and Microsoft simply ignored the discussion around the proposal and implemented their own version of the protocol. Launched in 1996, both Navigator and Internet Explorer allowed both first- and third-party cookies to be placed by default.

13.2 USER CONTROL OVER TRACKING

At the same time that Montulli proposed a mechanism to track people's behaviour on and between websites, other members on the mailing list discussed ways to preserve the privacy and control of the user. In Behlendorf's original call for a way to save website visitors' clickstreams, he also said that "any proposed solution *must* protect the anonymity of the user, for it's not really necessary to lose that when all that's care about is unique sessions." To Montulli's proposal, he responded that "[t]he browser should allow the user to NULL their [cookie] at any time (or turn off the functionality, even if by default it is on)". Koen Holtman, at the time a Master student at the Technical University Eindhoven in the Netherlands, suggested that browsers had some responsibility as well,[2] and proposed a control interface:

```
It is suggested that browsers provide something like the following
preferences box:
   +-------------------------------------------------------------------+
    Honor set-cookie requests:
        ( ) Always honor request
        ( ) Start honoring requests if one is done in a response to
            a form submission (POST request).
        (*) Ask once for every site, use reply in later sessions
        ( ) Never honor requests
   +-------------------------------------------------------------------+
   where the (*) is the default setting.
```

Version 1.1 of Netscape – released in April 1995 – did not offer users any control over the cookies: they could not set any preferences, were not notified when one was placed, and were all accepted by default. Netscape 2.0, released in September 1995, supposedly limited each website to placing only three cookies, although researchers trying to reconstruct this were able to have the browser accept four cookies in version 2.02.[3] It was possible, however, to limit the placement of cookies by editing attributes of the cookies.txt file (ignoring the "do not edit" comment).[4] Internet Explorer and Navigator – both released in August 1996, included a rudimentary version of the preference box suggested by Holtman. The browsers gave users the option to either accept all cookies (the default option)

[2] Koen Holtman (Aug. 11, 1995). *Non-persistent Cookie proposal*. www-talk electronic mailing list message. URL: https://lists.w3.org/Archives/Public/www-talk/msg01499.html.

[3] Lynette I Millett, Batya Friedman, and Edward Felten (2001). 'Cookies and web browser design: Toward realizing informed consent online.' In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 46–52.

[4] Ibid.

or be alerted when a site tried to place a cookie and choose in the moment. The alert would tell the user what server wanted to place the cookie, which server could use it in the future, the cookie ID, the cookie content, and its expiration date (fig. 41 via[5]).

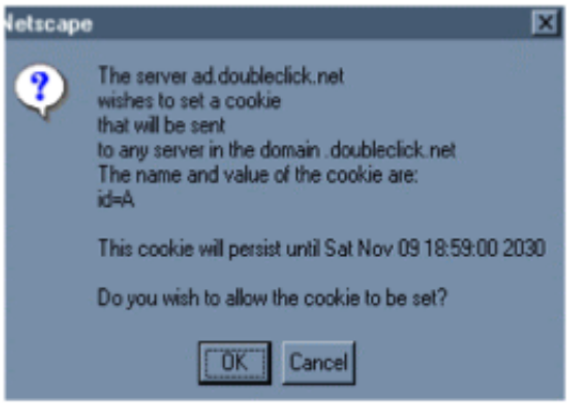


Figure 41. Cookie pop-up from Netscape 3.04.[6]

At least as early as 1996, protection against cookies also became an intermediate software market, with popular solutions such as Luckman's Anonymous Cookie for Internet Privacy, Kevin McAleavey's NSClean (fig. 42), or Limitsoft's Cookie Crusher.[7] These could automatically delete or reject cookies, prompt the user for a decision, and set guidelines for future choices on the same site.

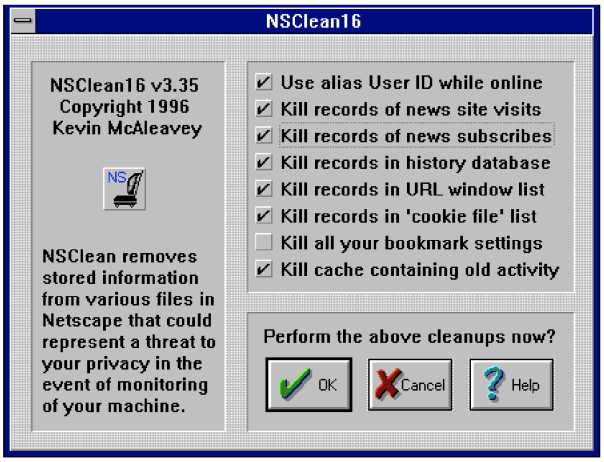


Figure 42. Online privacy management tool NSClean for NetScape, by Kevin McAleavey

The W3C also weighed in on the discussion and in May 1997[8] launched the Privacy Preference Project (P3P) to develop a technical specification for automated privacy decisions: the user could indicate what type of data they did or did not

[5] Millett, Friedman, and Felten, 'Cookies and web browser design: Toward realizing informed consent online.'

[7] Monique R. Brown (1998). 'Revenge of the cookie monster.' In: *Black Enterprise*, pp. 42–44.

[8] Joseph Reagle Jr. Lorrie Faith Cranor (1997). 'Designing a Social Protocol: Lessons Learned from the Platform for Privacy Preferences.' In: *Proceedings of the Telecommunications Policy Research Conference*.

want to share through their browser and, when visiting a website, the server would configure itself to comply with these preferences. The specification represented a huge effort from researchers, browser vendors, and industry and finally became a specification in 2002. Mozilla and Internet Explorer implemented support for the specification but almost no websites used it, and the protocol was declared obsolete in 2018.

13.3 REGULATORY RESPONSE

As digital data collection and surveillance technologies grew in quantity and scope, governmental authorities in both the US and the EU were forced to respond. This also revealed their ideological perspectives on the role of the state and how to best shape a society.

13.3.1 *The United States and Self-Regulation*

In the United States, the Clinton administration continued and further expanded the idea that data protection should be managed through the self-regulation of companies, rather than strong intervention of the state.[9] Their approach was instrumental in transforming computer *users* into computer *consumers* in the 1990s: privacy and data protection became a matter of consumer choice between competing offers, rather than decommodified, fundamental rights of citizens.[10] For example, it was the Federal Trace Commission, the agency tasked with upholding competition law and consumer protection, that started an investigation into DoubleClick in 1999 for planning to merge with Abacus Direct, which held a large database of consumer-purchasing data. This (and some of the ensuing legal battles DoubleClick became mired in) resulted not in data protection legislation to ensure such aggregation of citizen's data would only happen within certain constraints, but instead in DoubleClick launching extensive campaigns to educate users about how cookies worked as well as the creation of the "industry choice page for consumers", a central opt-out page accessible off-site after clicking through a website's multiple privacy policy pages. This preference for self-regulation has continued until today, although the adoption of stronger privacy and data protection regulations in other parts of the world have pressured the US to reconsider its position, and individual states are beginning to introduce legal instruments.

[9] Meghan Grosse (2020). 'Laying the foundation for a commercialized internet: international internet governance in the 1990s.' In: *Internet Histories* 0.0, pp. 1–16. DOI: 10.1080/24701475.2020.1769890. eprint: https://doi.org/10.1080/24701475.2020.1769890. URL: https://doi.org/10.1080/24701475.2020.1769890.

[10] Meg Leta Jones (2020). 'Surveillance Capitalism Online: Cookies, Notice Choice, and Web Privacy.' In: *Surveillance Capitalism in America: From Slavery to Social Media*. Ed. by Josh Lauer and Kenneth Lipartito. University of Pennsylvania Press, p. 23.

13.3.2 *The European Union and Government Regulation*

The European Union, instead, took a normative and rights-based approach to people's data and privacy.

13.3.2.1 *Data Protection Directive*

It adopted the Data Protection Directive (DPD) in 1995, which regulated the processing of data that could reveal an individual's identity – such as their name, address, credit card numbers, criminal record, etc – and the free movement of that data between EU Member States and other countries.[11] The motivation for the Directive came from the belief that European integration was endangered because of the diverging ways that Member States protected their citizens with respect to the processing of their data.[12] Unifying these, they believed, would better protect the fundamental right of a private life enshrined in article 8 of the European Convention on Human Rights, and help with the objective to establish an internal European market. The DPD allowed six ways in which a data controller could lawfully process an individual's data: (a) based on their consent; (b) for the purpose of a contract; (c) to comply with other laws; (d) because it would be in the data subject's vital interest; (e) to serve the public interest; (f) or for the legitimate interest of the data controller. In spite of its unifying goal, the DPD was a Directive, which meant that each EU Member State was allowed some liberties of interpretation when they transposed it into national law (compared to the literal translations required by Regulations). This resulted in a slightly deviating patchwork of data protection regulation across the continent. In terms of cookies and other web-tracking technologies, the DPD did not really directly affect their designs or the control mechanisms available to the user. Most Member States allowed cookies to be processed as part of the legitimate commercial interest of the data controller, which has no bearing on interface design.

13.3.2.2 *ePrivacy Directive*

The first EU regulation that has a significant effect on the design and use of cookies and adjacent software was the 2002 ePrivacy Directive (colloquially known as the Cookie Law). The focus of this directive was not on *personal* data, but instead on confidentiality of communication. It legislated the reading and writing of information on an individual's devices, of which cookies were just a single example. Described in in Article 5(3), it set out that storing data on or accessing data from a user's device is only allowed on the condition that the user "is provided with clear and comprehensive information in accordance with Directive 95/46/EC [...] about the purpose of the processing and is offered the right to refuse". It is in the reference to Directive 95/46/EC (the DPD) that interface design first becomes

[11] '95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data' (1995). In: *Official Journal of the EC* 23.6.

[12] Gloria González Fuster (2014). *The emergence of personal data protection as a fundamental right of the EU*. Vol. 16. Springer Science & Business, p.125.

important for legal compliance. Article 10 of the DPD specifies that a data subject should have information about the identity of the controller, the purpose for the processing, and any other information necessary "to guarantee fair processing".[13] And so, websites and advertising agencies invented the cookie pop-up to meet these requirements and continue to collect user's data.

The directive originally also stipulated that "prior, explicit" consent was required, but this provision was lobbied out by advertising organisations, who argued that it would destroy commerce on the internet.[14] This changed when in 2005 Mark Russinovich – a security expert – published a blog post detailing how Sony was installing rootkits on people's devices when they inserted one of their CDs.[15] The rootkit modified the operating system to prevent them from being copied, but also opened it up to vulnerabilities for other malware. The scandal and public uproar gave the European Commission enough momentum to bring back the concept of consent. It amended Article 5(3) of the ePrivacy Directive to say that, in addition to clear and comprehensive information, the lawful reading and writing of information on a user's device now also required that the user "has given his or her consent". To define consent, as with the specification for information, the amendment pointed to the Data Protection Directive, which defined in article 2(h) that consent needed to be a "freely given, specific, and informed". Consequently, the design of cookie pop-ups needed to change.

Although pop-up interfaces were not the only way that consent could be given, it was the recommended method. Recital 17 of the ePrivacy Directive states that consent can be given in a number of different ways, through browser settings or by ticking a box when visiting a website. However, A29WP criticised a browser based approach, because they argued that it would muddy the concept of consent, that many browsers' cookie settings did not meet the requirements of the regulation, and that certain cookies (e.g., Flash cookies) could not even be controlled via the browser. They state that a pop-up window would be an adequate way of meeting the requirements for informed consent.[16] And so, the original information-providing cookie banners from 2002 now got an 'I consent' button added to it (fig. 43).

The ePrivacy Directive is generally considered a regulatory failure, as most websites purposefully flouted or simply ignored the requirements for legally valid consent and information (for an analysis of the Dutch case, see[17]).

[13] '95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.'

[14] Sylvia Mercado Kierkegaard (2005). 'How the cookies (almost) crumbled: Privacy & lobbyism.' In: *Computer Law & Security Review* 21.4, pp. 310–322.

[15] Eleni Kosta (2013a). 'Peeking into the cookie jar: the European approach towards the regulation of cookies.' In: *International journal of law and information technology* 21.4, pp. 380–406.

[16] Article 29 Working Party (2009). *OPINION 1/2009 on the proposals amending Directive 2002/58 on privacy and electronic communications (e-Privacy Directive), WP159*. European Commission.

[17] Ronald Leenes and Eleni Kosta (2015). 'Taming the cookie monster with dutch law–a tale of regulatory failure.' In: *Computer Law & Security Review* 31.3, pp. 317–335.

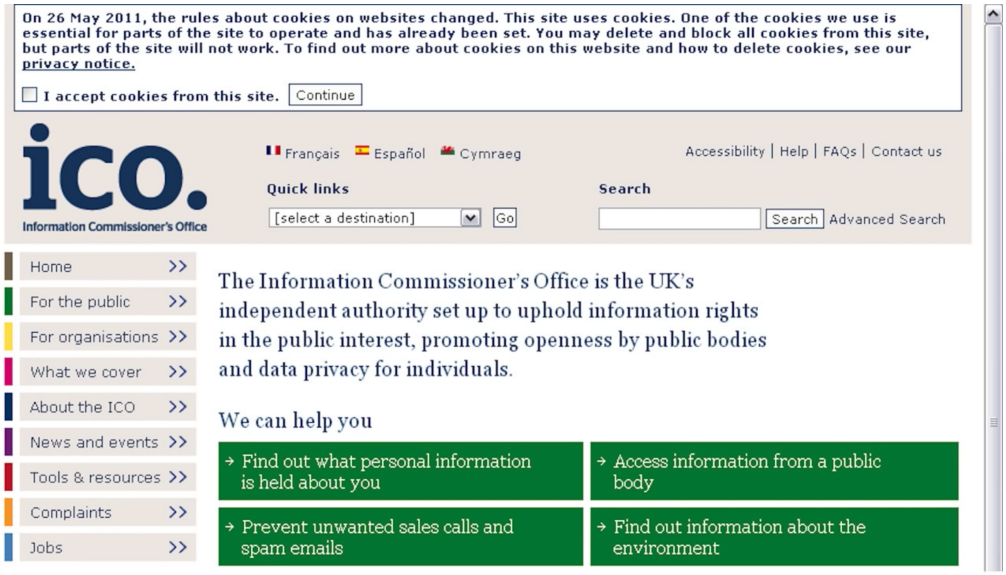


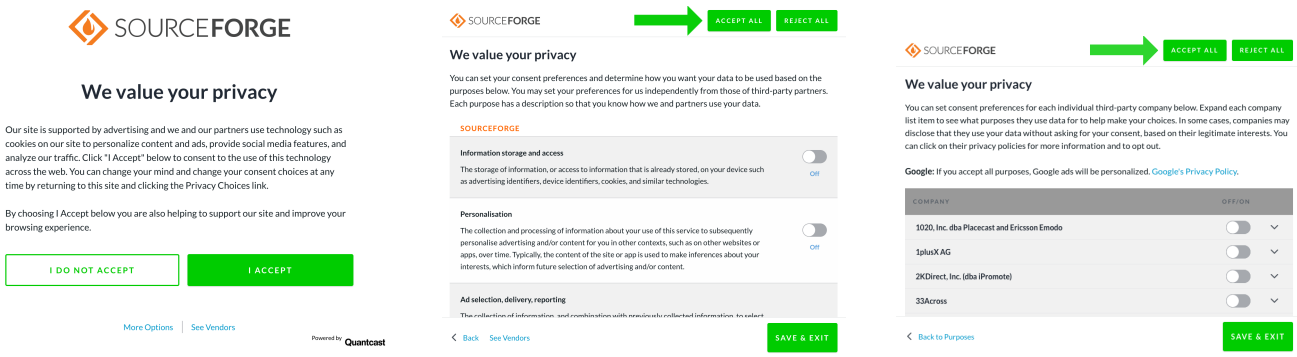
Figure 43. The cookie pop-up of the Information Commissioner's Office in 2011, the UK's independent authority for data protection and privacy.

13.3.3 *The General Data Protection Regulation*

The Data Protection Directive was conceived and implemented at a time when the internet and data processing were marginal concerns, rather than a fundamental aspect of European societies and economies. As privacy and data collection scandals continued to occur with predictable regularity, originating both from governments and commercial companies, the European Commission began working on a reform of the DPD in 2012. The goal was to provide individuals with more rights and control over their personal information, while directing the flow of data between countries and companies through procedural requirements that would promote more responsible use: an ambition largely identical to the motivation for the DPD. Four years later, in May 2016, and significantly behind the imagined schedule (with the Snowden revelations in the midle of it), the General Data Protection Regulation (GDPR) was adopted. Two years after that, in 2018, it finally went into effect.

The GDPR is a comprehensive piece of legislation whose consequences are still playing out, but in relation to cookies and consent mechanisms it matters mostly only in relation to the ePrivacy Directive. The GDPR repeals the DPD, which means that any other legislation referring to definitions in the latter, are now automatically referred to the new definitions of the GDPR. For the reading and writing of cookies, this means that the requirements of consent and information are significantly enhanced by the more stringent requirements of the GDPR, which defined consent as "any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her".[18] As a result, the old cookie consent interfaces no longer

[18] European Union (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal*



(a) First page **(b)** Categories and purposes **(c)** Vendors/third-parties

Figure 44. The three components of the QuantCast CMP on https://sourceforge.net in September 2019.

sufficed, and new designs needed to be made. Quickly enough, third-party solutions emerged that purported to help websites achieve legal compliance through something called a Consent Management Platform (CMP) (fig. 45).

More sophisticated than the first banners of the early 2000s, these platforms allowed websites to connect the "consent" they acquired from visitors to the real-time advertising bidding infrastructure that underlies the vast majority of data processing on the internet these days. With such vested interests, the design of these CMPs often use (illegal) dark patterns: design decisions which exploit human psychology to guide the decisions made by users. At the time of writing, roughly a quarter of websites is using third-party CMPs, and this rate continuous to go up.[19]

13.4 CONCLUSION

There is much more to web-based tracking, consent software, and regulation than what is described in this chapter. Beyond cookies, websites use technologies such as invisible tracking pixels and canvas fingerprinting based on how a device renders HTML elements that are not necessarily covered by the ePrivacy Directive. Many browser extensions have emerged that try to fight back using blocklists of known tracker endpoints. More protocol innovations have been suggested, such as Do Not Track, that could send signals to webservers about an individual's privacy preferences, and browsers have started to phase out third-party cookies completely. The EU is working on repealing the ePrivacy Directive and a number of states in the US have implemented their own regulatory responses. But while the details might have changed, the position of the main chess pieces have stayed pretty much the same since the early 1990s. Commercial companies continue to track individuals across the internet to make money; the United States government still believes self-regulation could work; the European Union continous

data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1.

[19] Adzerk (2019). *Adtech Insights — August 2019 Report*. URL: https://adzerk.com/assets/reports/AdTechInsights_Aug2019.pdf.

down its path of being a normative regulatory force on the global digital stage; and users of the internet continue to have little control over their personal data and digital technologies.

14.

DARK PATTERNS AFTER THE GDPR: SCRAPING CONSENT POP-UPS AND DEMONSTRATING THEIR INFLUENCE

Based on Nouwens, Liccardi, Veale, Karger, and Kagal[1]

14.1 INTRODUCTION

The predominant method of giving people some semblance of control over their privacy while browsing the web is 'notice and choice' or 'notice and consent'.[2] These mechanisms involve showing an individual an informational statement and, depending on their (in)action, acquiring or assuming their agreement to collecting, storing, and processing their data. To many, this practice has become informally known as 'cookie banners'.

What counts as sufficient notice, and what counts as legally-acceptable consent, significantly differs depending on the geographical and regulatory scope that an actor falls in. The application in Europe of the General Data Protection Regulation (GDPR)[3] from May 2018, together with recent regulatory guidance from data protection authorities (DPAs) and jurisprudence from the Court of Justice of the European Union (CJEU), has highlighted the illegality of the way 'notice and consent' has hitherto functioned in the EU. These regulatory changes have both clarified the concept of consent in European law, as well as brought more significant (and extraterritorial) consequences for flaunting these rules. EU law in particular focuses on the *quality* of the consent required, and its freely-given, optional nature.

[1] Midas Nouwens et al. (2020b). 'Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376321. URL: https://doi.org/10.1145/3313831.3376321.

[2] Lorrie Faith Cranor (2012). 'Necessary but Not Sufficient: Standardized Mechanisms for Privacy Notice and Choice The Economics of Privacy.' In: *Journal on Telecommunications and High Technology Law* 10.2, pp. 273–308.

[3] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*.

Consent management platforms (CMPs) have gained traction on the Web to help website owners outsource regulatory compliance. These (often third-party) code libraries purport to help websites establish a lawful basis to both read and write information to users' browsers and to process these individuals' personal data, often for the purposes of tracking and complex advertising transactions, such as 'real-time bidding'.[4]

This intertwining of interface designs and data protection and privacy law raises significant questions. This chapter deals with two of them:

1. What is the current state of interface design of CMPs in the EU, and how prevalent are non-compliant design elements?

2. How do interface designs affect consent actions of users and, by extension, how 'freely given' that consent is?

To answer the first question, we surveyed the designs of the 5 most commonly used third-party CMPs by scraping their varied implementations on the top 10,000 most popular websites in the United Kingdom (UK) (n=680); and evaluated them against European law and regulatory guidance. To answer the second question, we built a browser plugin that injects consent notices into webpages and ran a controlled experiment (n=40) with eight different interfaces to see how they affect participants' consent responses.

14.2 CONSENT AND WEB TECHNOLOGIES UNDER EU LAW

EU law considers users' devices and information within them part of their private sphere. Relevant protection is extended to all EU residents and to all individuals around the world being delivered online services from the Union. In light of a growing trend in the early 2000s of rightsholders sneaking piracy-spotting rootkits onto users' devices,[5] the ePrivacy Directive[6] was amended to require that storing or accessing information on a user's device not 'strictly necessary' for providing an explicitly requested service requires both clear and comprehensive information and opt-in consent.[7] This also applies to cookies, HTML web storage, and fingerprinting in browsers providing non-essential features such as tracking. Such consent is however, *not* required for essential functions such as remembering login status, a shopping cart, or cookies for data security required by law.[8]

[4] Information Commissioner's Office (June 2019b). *Update Report into Adtech and Real Time Bidding*. Wilmslow, Cheshire: ICO.

[5] Eleni Kosta (2013b). 'Peeking into the Cookie Jar: The European Approach towards the Regulation of Cookies.' In: *International Journal of Law and Information Technology* 21.4, pp. 380–406. DOI: 10.1093/ijlit/eat011.

[6] European Union (2002). *Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications) OJ L 201*.

[7] Kosta, 'Peeking into the Cookie Jar.'

[8] Commission nationale de l'informatique et des libertés (CNIL) (2019). *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*; Information Commissioner's Office (July 2019a). *Guidance on the Use of Cookies and Similar Technologies*. Wilmslow, Cheshire: ICO.

The ePrivacy Directive is connected to definitions in European data protection law, so when the GDPR[9] repealed and replaced the Data Protection Directive 1995[10] in 2018, these practices became subject to new, heightened standards concerning the quality of consent. The GDPR defines consent as "any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her".[11] Several aspects of this legal regime with design implications are important to highlight here, which are drawn from the legal texts, regulators' guidance, and court cases or opinions.

14.2.1 *Freely given and unambiguous consent*

Regulators and the Court of Justice of the EU have both emphasised that for consent to be freely given and informed, it must be a separate action from the activity the user is pursuing.[12] So-called 'implicit' or 'opt-out' consent — continuing to use a website without active objection to a notice — is not a clear positive action and as such will not establish a valid legal basis to lay cookies or process data on the basis of consent.[13]

As a consequence of the importance of the freely given nature of consent, design matters for legal compliance. Pre-ticked boxes, which require a positive action to opt-out from, are explicitly singled out in the GDPR as an invalid form of consent.[14] The Court of Justice has recently ruled that they were also not a valid

[9] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*.

[10] European Union (1995). *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ 1995 L 281/31*.

[11] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, art 4(11).

[12] Advocate General Szupunar (2019). *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:246, Opinion of the Advocate General*; Article 29 Working Party (2018). *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*. European Union; Court of Justice of the European Union (2019b). *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:801*.

[13] Article 29 Working Party, *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*; Commission nationale de l'informatique et des libertés (CNIL), *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*; Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

[14] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, recital 32.

form of consent under the previous law, operational since the mid-90s.[15] The UK's Information Commissioner's Office further states that "[a] consent mechanism that emphasises 'agree' or 'allow' over 'reject' or 'block' represents a non-compliant approach, as the online service is influencing users towards the 'accept' option." Similarly, cookie boxes without a 'reject' option, or where it is located in a 'more information' section or on a third party webpage, are also non-compliant.[16] One of the CJEU's Advocates General (official impartial advisors to the Court on cases raising new points of law) has emphasised the need that both actions, "optically in particular, be presented on an equal footing".[17]

Moreover, it must be "as easy to withdraw as to give consent".[18] This means if consent was gathered through "only one mouse-click, swipe of keystroke", withdrawal must be "equally as easy" and "without detriment" or "lowering service levels".[19]

An issue of continued contention is the validity of so-called 'cookie walls', whereby consent is a prerequisite to accessing a website. While several regulators appear minded to suggest in many or all cases this practice is illegal,[20] the issue remains unclear[21] and the final conclusion will regardless be subject to the "glacial flow" of the draft ePrivacy Regulation through the EU's legislative process.[22]

14.2.2 *Specific and informed consent*

An important aspect of data protection is *purpose limitation*, meaning users must consent in relation to a particular and specific purpose for processing data. They cannot provide *carte blanche* for a data controller to do whatever they like.[23] These purposes cannot be inextricably 'bundled', so an 'accept all' button is only

[15] Court of Justice of the European Union, *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:801*.

[16] Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

[17] Advocate General Szupunar, *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:246, Opinion of the Advocate General*, para 66.

[18] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, art 7(3).

[19] Article 29 Working Party, *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*.

[20] Autoriteit Persoonsgegevens (2019). *Hoe Legt de AP de Juridische Normen Rond Cookiewalls Uit?* Den Haag: AP; Commission nationale de l'informatique et des libertés (CNIL), *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*; European Data Protection Supervisor. *EDPS Opinion on the Proposal for a Regulation on Privacy and Electronic Communications (ePrivacy Regulation), Opinion 6/2017*. Brussels, BE: EDPS; Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

[21] Frederik J Zuiderveen Borgesius et al. (2017). 'Tracking Walls, Take-It-Or-Leave-It Choices, the GDPR, and the ePrivacy Regulation.' In: *European Data Protection Law Review* 3.3, pp. 353–368. DOI: 10/gfsh4x.

[22] Oisin Tobin (2019). 'Cookie consent revisited.' In: *Privacy and Data Protection* 19 (5), p. 11.

[23] As an unalienable fundamental right, it is impossible for an EU resident to 'sign away' their right to effective data protection.

compliant if it is additional to the possibility of specifically consenting to each purpose.[24]

Furthermore, consent is invalid unless all organisations processing this data are specifically named.[25] Simply linking to an external list of potential vendors, which may not represent the code being run on the linking webpage, is "insufficient to provide for free and informed consent".[26] Consent should be able to be rejected at the same level as the 'accept' button, so having to navigate further to third party websites to reject tracking is non-compliant.[27] Information required to be provided to data subjects includes certain GDPR–mandated information (including controller contact, processing purposes, legal basis, recipients and sources of data, international transfers, storage period, data rights and rights of complaint, and meaningful information about the logic of significant automated decision-making),[28] as well as the duration of cookies.[29]

14.2.3 *Efficient and timely data protection*

Individuals have the right to 'efficient and timely' protection of their data rights, meaning where consent is required, it is required prior to data processing, not subsequently.[30] Cookies must not be set before the user has expressed their affirmative consent. Furthermore, fresh consent is required when new, non-essential cookies are being set by a new third party.[31] The burden is on the data controller to be able to demonstrate that they adhere to data protection law and principles, including that they have valid consent for each individual.[32]

---

[24] Commission nationale de l'informatique et des libertés (CNIL), *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*.

[25] Commission nationale de l'informatique et des libertés (CNIL), *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*; Information Commissioner's Office, *Update Report into Adtech and Real Time Bidding*.

[26] Information Commissioner's Office, *Update Report into Adtech and Real Time Bidding*.

[27] Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

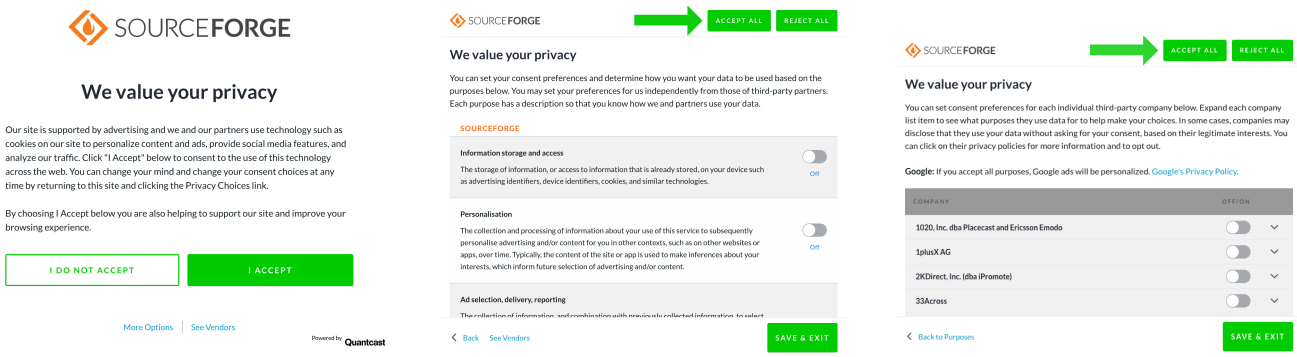
[28] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, arts 13–14.

[29] Advocate General Szupunar, *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:246, Opinion of the Advocate General*; Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

[30] Article 29 Working Party, *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*; Court of Justice of the European Union (2019a). *Case C-49/17* Fashion ID GmbH & Co.KG v Verbraucherzentrale NRW eV. *ECLI:EU:C:2019:629*.

[31] Article 29 Working Party, *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*; Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

[32] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, art 5(2).



(a) First page **(b)** Categories and purposes **(c)** Vendors/third-parties

Figure 45. The three components of the QuantCast CMP on https://sourceforge.net in September 2019.

14.3 RELATED WORK

14.3.1 *Notice & Consent*

The predominant model for communicating information privacy protections to end-users has been notice(/awareness) and consent(/choice). The interface designs of this model have mostly been privacy policies and opt-in/out interfaces,[33] which legally can be seen as "pre-formulated declarations of consent", or "click-wrap" contracts.[34] The usability challenges of these interfaces have seen considerable work across disciplines, largely divisible in studies that establish the shortcomings of interface designs, and studies proposing alternative technologies. Privacy policy notices are notorious for taking a disproportionate amount of time to go through and require reading comprehension abilities at university level.[35] Privacy policies are rarely read by users[36] prior to using or visiting a site/service. Users have been shown to (almost automatically) consent without viewing them[37] since they stand in the way of the users' primary goal: accessing the ser-

[33] Cranor, 'Necessary but Not Sufficient.'

[34] Damian Clifford, Inge Graef, and Peggy Valcke (2019). 'Pre-formulated Declarations of Data Subject Consent—Citizen-Consumer Empowerment and the Alignment of Data, Consumer and Competition Law Protections.' In: *German Law Journal* 20.5, pp. 679–721.

[35] Carlos Jensen and Colin Potts (2004). 'Privacy policies as decision-making tools: an evaluation of online privacy notices.' In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, pp. 471–478.

[36] H. Nissenbaum (2011). 'A contextual approach to privacy online.' In: *Daedalus* 140.4, pp. 32–48; Jonathan A. Obar and Anne Oeldorf-Hirsch (2018). 'The biggest lie on the Internet: ignoring the privacy policies and terms of service policies of social networking services.' In: *Information, Communication & Society* 0.0, pp. 1–20. DOI: 10.1080/1369118X.2018.1486870. eprint: https://doi.org/10.1080/1369118X.2018.1486870. URL: https://doi.org/10.1080/1369118X.2018.1486870; Tony Vila, Rachel Greenstadt, and David Molnar (2003). 'Why We Can'T Be Bothered to Read Privacy Policies Models of Privacy Economics As a Lemons Market.' In: *Proceedings of the 5th International Conference on Electronic Commerce*. ICEC '03, pp. 403–407.

[37] Alessandro Acquisti and Jens Grossklags (2005). 'Privacy and rationality in individual decision making.' In: *Security Privacy, IEEE* 3.1, pp. 26–33; Julio Angulo et al. (2011). 'Towards Usable Privacy Policy Display & Management for PrimeLife.' In: *S. M. Furnell, & N. L. Clarke (Eds.), Proceedings of international symposium on human aspects of information security & assurance (HAISA*

vice.[38] This behaviour has been attributed to the users' difficulty understanding how to make meaningful decisions about their privacy preferences; but even in situations where they are made aware of the implications of their decision, they prefer short-term benefits over long-term privacy.[39] Because of this, control mechanisms of these notices are considered illusory in practice[40] — sometimes having devolved into merely an informational statement rather than an interactive control panel.

The perceived ineffectiveness of this approach has given rise to a number of design alternatives (for an overview of the entire design space, see[41]). Gage Kelley et al. proposed standardised "nutrition label" notices with icons representing the type of data collected and how it is used, and showed how it helped users find information more quickly and accurately.[42] Reeder et al. developed an interactive matrix visualisation called Expandable Grid which shows a colour-coded overview of a policy that can be expanded for more detail.[43] The Platform for Privacy Preferences (P3P) was an involved attempt to help automate some of this process by building a machine-readable language for expressing website privacy policies which could then interface with user agents, such as the browser or other privacy applications.[44] While it was implemented by Microsoft for Internet Explorer and Edge, P3P never achieved widespread adoption, partly because its comprehensiveness was seen as too complex for regular website owners to apply but also because there was no regulatory or political impetus to force browser vendors to use it.

The majority of studies around notice and consent have focused on how well the interface design helps users make informed decisions. This paper focuses more on the legal *quality* of the consent that is collected.

14.3.2 *Dark patterns*

Interface designs that try to guide end-users into desired behaviour through malicious interaction flows are referred to as "dark patterns".[45] As a phenomenon

2011), pp. 108 –117; Meinert David B. et al. (2006). 'Towards Usable Privacy Policy Display & Management for PrimeLife.' In: *Journal of Electronic Commerce in Organizations (JECO)* 4.1, pp. 1–17; A. M. McDonald and L. F. Cranor (2008). 'The cost of reading privacy policies.' In: *I/S: A Journal of Law and Policy for the Information Society* 4, pp. 540 –565; Nissenbaum, 'A contextual approach to privacy online.'

[38] Acquisti and Grossklags, 'Privacy and rationality in individual decision making'; Angulo et al., 'Towards Usable Privacy Policy Display & Management for PrimeLife.'

[39] Acquisti and Grossklags, 'Privacy and rationality in individual decision making.'

[40] Fred H Cate (2010). 'The limits of notice and choice.' In: *IEEE Security & Privacy* 8.2, pp. 59–62.

[41] Florian Schaub et al. (2015). 'A design space for effective privacy notices.' In: *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pp. 1–17.

[42] Patrick Gage Kelley et al. (2009). 'A nutrition label for privacy.' In: *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, p. 4.

[43] Robert W Reeder et al. (2008). 'Expandable grids for visualizing and authoring computer security policies.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1473–1482.

[44] Lorrie Cranor (2002). *Web privacy with P3P*. Sebastopol, CA: O'Reilly Media.

[45] Colin M Gray et al. (2018). 'The dark (patterns) side of UX design.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 534.

they are part of the larger research agenda around persuasive design[46] and nudging.[47] The practice of dark patterns for privacy notices — while only sometimes discussed under this moniker in HCI and privacy literature[48] — is extensively reported on by consumer protection organisations,[49] white papers,[50] and popular press[51] (for an excellent overview, see the Norwegian Forbrukerrådet document "Deceived by Design"[52]). Its infamy has led the European Union and data protection officers to specifically highlight certain common dark patterns as non-compliant examples of the GDPR in its advisory documents such as privacy intrusive default settings, hiding away privacy-friendly choices and requiring more effort from the user to select it, illusory or take-it-or-leave-it choices, etc. Senators from the United States have recently introduced a draft bill specifically aimed at outlawing such practices, stating that it should be prohibited for any large online operator to "design, modify, or manipulate a user interface with the purpose or substantial effect of obscuring, subverting, or impairing user autonomy, decision-making, or choice to obtain consent or user data".[53]

Since the submission of this paper, two studies have been released that look specifically at the consent management platforms that have appeared in response to the GDPR.

Utz et al[54] analysed a random sample of 1,000 CMPs and manually categorised them along different design dimensions (e.g., positioning, size, consent options). They found (among other things) that a minimum of 57.4% used dark patterns to nudge users to select privacy-unfriendly options, and that 95.8% provide either

[46] Brian J Fogg (2009). 'A behavior model for persuasive design.' In: *Proceedings of the 4th international Conference on Persuasive Technology*. ACM, p. 40.

[47] Alessandro Acquisti et al. (Aug. 2017). 'Nudges for Privacy and Security: Understanding and Assisting Users' Choices Online.' In: *ACM Comput. Surv.* 50.3, 44:1–44:41. ISSN: 0360-0300. DOI: 10.1145/3054926. URL: http://doi.acm.org/10.1145/3054926; Richard H Thaler and Cass R Sunstein (2009). *Nudge: Improving decisions about health, wealth, and happiness*. Penguin.

[48] Christoph Bösch et al. (2016). 'Tales from the dark side: Privacy dark strategies and privacy dark patterns.' In: *Proceedings on Privacy Enhancing Technologies* 2016.4, pp. 237–254; Gregory Conti and Edward Sobiesk (2010). 'Malicious Interface Design: Exploiting the User.' In: *Proceedings of the 19th International Conference on World Wide Web*. ACM, pp. 271–280; Gray et al., 'The dark (patterns) side of UX design'; Arunesh Mathur et al. (2019). 'Dark patterns at scale: Findings from a crawl of 11K shopping websites.' In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW, p. 81.

[49] Forbrukerrådet (2019). *Deceived by Design: How tech companies use dark patterns to discourage us from exercising our rights to privacy*. URL: https://fil.forbrukerradet.no/wp-content/uploads/2018/06/2018-06-27-deceived-by-design-final.pdf.

[50] European Data Protection Supervisor (2018). 'EDPS Opinion on the legislative package "A New Deal for Consumers".' In: URL: https://edps.europa.eu/sites/edp/files/publication/18-10-05_opinion_consumer_law_en.pdf.

[51] Natasha Singer (May 2016). 'When Websites Won't Take No for an Answer.' In: *New York Times*. URL: https://www.nytimes.com/2016/05/15/technology/personaltech/when-websites-wont-take-no-for-an-answer.html?mcubz=0&_r=0.

[52] Forbrukerrådet, *Deceived by Design: How tech companies use dark patterns to discourage us from exercising our rights to privacy*.

[53] Mark R. Warner Deb Fisher (2019). 'Deceptive Experiences To Online Users Reduction (DETOUR) Act.' In: URL: https://www.scribd.com/document/405606873/Detour-Act-Final.

[54] Christine Utz et al. (2019). '(Un)Informed Consent: Studying GDPR Consent Notices in the Field.' In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: ACM, pp. 973–990. ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3354212. URL: http://doi.acm.org/10.1145/3319535.3354212.

no consent choice or confirmation only. They conducted a follow-up experiment to test the effects of the CMP position, the granularity and nudging of choices, and the technicality of the language and presence of a privacy policy link. They demonstrate that positioning the CMP in the lower (left) part of the screen increases interaction rates; users are more likely to accept tracking given a binary choice than when given more granular options; acceptance rate increased from a mere 0.16% to 83.55% when options were preselected; and technical language and privacy policies have a minor effect on consent choice.

The work by Matte, Bielova, and Santos[55] investigates the actual consent signal sent from the CMP to the respective data processors. They detect that 12.3% of 1,426 sites send a consent signal before the user makes a choice. Semi-automatically reviewing 560 sites reveals that 54% of them contain at least one violation regarding the way consent is determined, asked, or complied with.

14.3.3 *Empirical Studies of EU Privacy Regulation*

Various studies have tried to chart the impact of European privacy regulation on the collection and processing of personal data on the web, both within its territorial scope and globally. A longitudinal 4-year study of the impact of the revised ePrivacy directive on cookie placement shows that 1) 49% of websites placed cookies before receiving consent; 2) 28% of websites did not provide any consent mechanism; and 3) the percentage of websites violating the directive stayed constant over the course of 4 years, indicating the policy to be ineffective.[56]

With respect to the GDPR, both industry and academia have been monitoring its effects since being introduced in May 2018. Degeling et al.[57] monitored the prevalence of privacy policies on websites before and after the introduction of the regulation, showing that in some EU member states the number of policies increased by 15.7% (to a total of 84.5%), while 72.6% of sites updated documents they already had. They estimate that a total of 62.1% of websites in Europe display a consent notice, an increase of 16% since shortly before the regulation became enforceable. Adzerk, an ad tech company, places this percentage considerably lower, at a mere 20.4%,[58] although their methodology is more restrictive than Degeling et al.'s. Interestingly, QuantCast, one of the largest CMP providers, also brought out a report stating that over 90% of users (n=1bn) have consented to data processing.[59]

[55] Célestin Matte, Nataliia Bielova, and Cristiana Santos (2019a). 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.' In: Under submission. URL: https://arxiv.org/abs/1911.09964v1.

[56] Martino Trevisan et al. (2019). '4 Years of EU Cookie Law: Results and Lessons Learned.' In: *Proceedings on Privacy Enhancing Technologies* 2019.2, pp. 126–145.

[57] Martin Degeling et al. (2018). 'We Value Your Privacy... Now Take Some Cookies: Measuring the GDPR's Impact on Web Privacy.' In: *arXiv preprint arXiv:1808.05096*.

[58] Adzerk, *Adtech Insights — August 2019 Report*.

[59] John McCarthy (2019). *Over 90% of users consent to GDPR requests says Quantcast after enabling 1bn of them*. https://www.thedrum.com/news/2018/07/31/over-90-users-consent-gdpr-requests-says-quantcast-after-enabling-1bn-them.

CMP	Sites	Median vendors (low./upp. quartiles)	Explicit/implicit consent	Banner/barrier	Preticked options	Minimum compliance
Cookiebot	12.5% (85)	104 (61, 232)	45/40	78/7	64 (75.3%)	2 (5.6%)
Crownpeak	12.2% (83)	38.5 (18.8, 132.3)	46/37	52/31	67 (80.7%)	0 (0%)
OneTrust	24.3% (165)	58 (26.5, 104.5)	47/118	158/7	108 (65.4%)	3 (1.8%)
QuantCast	41% (279)	542 (542, 542)	279/0	132/147	90 (32.3%)	73 (26.2%)
TrustArc	10% (68)	87 (38, 152)	42/26	26/42	53 (77.9%)	2 (2.9%)
all	**680**	**315 (58, 542)**	**459/221**	**446/234**	**382 (56.2%)**	**80 (11.8%)**

Table 8. Key statistics on scraped CMPs.

Sanchez-Rola et al.[60] performed an evaluation of the tracking undertaken by 2,000 high-traffic websites and evaluated how information notices and actual tracking behaviour changed. They found that the GDPR affected EU and US sites in the same way, that consent management platforms reduced the amount of tracking, but that personal data collection is still ubiquitous: 90% still made use of cookies that were able to identify individual users. Sørensen and Sokol[61] present a more nuanced picture of the shifts in third-party tracker presence and behaviour, showing a decrease mostly present in private websites, whereas websites hosted by public institutions mostly stayed the same between February and September 2018. Along the same lines, there exists a difference between EU and non-EU private-sector sites, but little difference in public sites. Depending on the purpose category the tracker falls into, further distinctions can be made. The largest shift was visible in data collection for advertising, and the least in those used for cybersecurity. Overall, only 151 third-party trackers are used by 1% or more of the websites, while the remaining long-tail of 968 have a share of less than one percent.

14.4 STUDY 1: SCRAPING CMP INTERFACE DESIGNS

Little is known about many aspects of consent management platforms on the Web, particularly around the consent modalities, quality of this consent and related practices found in the field in the European Union. The major five CMP vendors offer a wide range of customisation options for their clients, and so from an identification of the CMP vendor it does not follow that many assumptions can be made about the interface design. To understand the status quo of consent management plaform interface design after the GDPR, we developed a web scraper

[60] Iskander Sanchez-Rola et al. (2019). 'Can I Opt Out Yet?: GDPR and the Global Illusion of Cookie Control.' In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS '19. Auckland, New Zealand: ACM, pp. 340–351. ISBN: 978-1-4503-6752-3. DOI: 10.1145/3321705.3329806. URL: http://doi.acm.org/10.1145/3321705.3329806.

[61] Jannick Sørensen and Sokol Kosta (2019). 'Before and After GDPR: The Changes in Third Party Presence at Public and Private European Websites.' In: *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: ACM, pp. 1590–1600. ISBN: 978-1-4503-6674-8. DOI: 10.1145/3308558.3313524. URL: http://doi.acm.org/10.1145/3308558.3313524.

to collect information about the five most commonly used third-party CMPs in the top 10,000 most-visited websites in the United Kingdom.

While their sophistication varies, surveyed CMPs all share similarities in back-end function. When a user accesses a site, the CMP detects their IP address and checks their cookies or local storage for any previously set consent preferences, and retrieves this data. If this fails, or if the CMP decides their preferences have expired, the user is shown a consent notice, and their response is recorded. This consent status is then passed on to any integrated tag firing rules, ad servers, and real-time bidding platforms the website has employed.

Visually, the CMP interfaces generally consist of three parts: 1) a first page describing the general purpose of the consent pop-up, with bulk consent options ('accept all' and, for some, 'reject all') (fig. 45a); 2) a second page with a more detailed description of the different data processing categories or purposes (e.g. personalisation, marketing), the ability to toggle them individually or collectively, and a button to submit the current consent state (fig. 45b); and, 3) a third page with a breakdown of all the vendors for whom the data is collected or with which it is shared, again with the ability to toggle individually or collectively, and a button to save these settings (fig. 45c). Not all deployed CMPs have all parts of these interfaces enabled.

14.4.1 *Method*

We built a Web scraper to collect data about the CMP's visual elements, interaction design, and text content (e.g. names of data processing categories or vendors). The scraper utilised the Python library *Scrapy*[62] and JavaScript rendering service *Splash*[63]. The variables the scraper collected included the CMP vendor, the notification style (banner, barrier, other), the type of consent (explicit or implicit) and specific user actions counted as consent (consent/visit/navigation/reloading/scrolling/closing the pop-up/clicking the page); the existence of accept and reject all buttons and the minimum number of clicks to make them available; for both vendors and categories/purposes, the existence of lists of these, their extent and descriptions, whether or which are enabled for user control, and their default state(s).

We ran the scraper from a Danish IP address[64] over 3 days in September 2019 over the top 10,000 UK sites according to webtraffic service *Alexa*. We throttled our scraper to two concurrent URL requests and no concurrent requests per domain, with a delay of 2 seconds. We cycled through three different user agents copied from our browsers to make sure the websites treated us as normal visitors, rather than an automated crawler. The CMPs the scraper was designed for are third-party services as identified by *Adzerk* in August,[65] which together account for ~58% of the market share: QuantCast, OneTrust, TrustArc, Cookiebot, and

[62] https://github.com/scrapy/scrapy

[63] https://github.com/scrapy-plugins/scrapy-splash

[64] Relevant legislation is harmonised across the EU and so a Danish IP and UK IP are the same jurisdiction for our purposes.

[65] A company that does server-side ad serving and writes reports about the state of the industry: www.adzerk.com

Crownpeak. We targeted UK sites, rather than sites across all EU countries, because the Adzerk report gives us information about the total population of CMPs in the UK market. This allowed us to check that our scraper's sample was representative both in number of CMPs identified and the overall distribution of the five most popular ones.

To determine the presence of a particular CMP, the scraper looked for an identifying HTML element within 5–15 seconds of arriving on the site (depending on the particular CMP and how it injects the pop-up). Data to construct the variables were extracted by querying for elements and attributes, traversing the DOM if no unique indentifiers existed, or accessing globally scoped objects. This data was pushed to a *MongoDB* database. Before deployment, the data returned by the scraper was manually validated with 40 randomly selected sites from the list of 10,000 for each of the five CMPs.

14.4.2 *Understanding compliance*

Based on the above section on EU law, we consider three core, measurable conditions that providers will have to meet to be considered legally compliant for the purpose of this study. This serves as a minimum hurdle: meeting these conditions alone will not guarantee compliance with the law, as there are a multitude of aspects and provisions, many of which can only be appropriately assessed qualitatively. However, these are conditions that are testable with the variables from our scraper, and therefore provide a window on the *maximum* level of compliance in the industry today. These conditions are:

CONSENT MUST BE EXPLICIT This condition is true if consent is a clear, positive, affirmative act, such as clicking a button, rather than e.g. continuing to navigate a website.

ACCEPTING ALL IS AS EASY AS REJECTING ALL Consent must be as easy to give as to withdraw/refuse. This condition is met if accepting all takes the same number of clicks as rejecting all, and automatically not met in the case where consent requires no clicks (i.e. Condition 1 is violated)

NO PRE-TICKED BOXES Consent to any vendor or purpose must be through affirmative acts at all granularity. If no non-necessary purposes or vendors are automatically on, this condition is met.

Factors which could contribute to non-compliance which we did *not* examine include qualitatively considering the information provided (e.g. specificity of purposes, contact details of vendors, provision of the duration of cookies), nor certain visual features such as colour or size or prominence of buttons beyond clicks.

14.4.3 *Results*

680 (6.8%) of the top 10,000 UK websites contained a CMP which could be successfully scraped by our tool. According to a survey of the top 10K UK websites

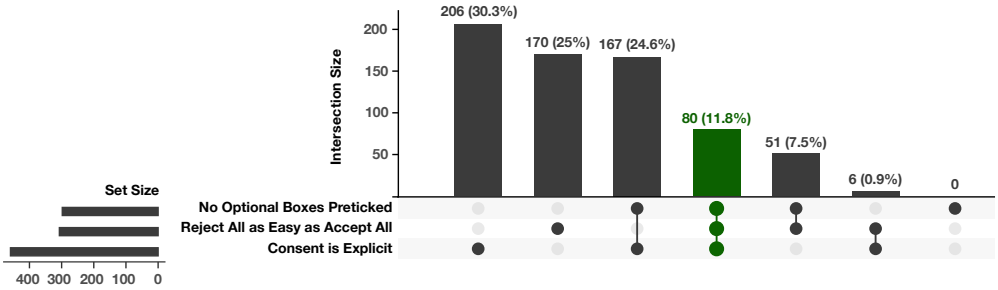


Figure 46. UpSet diagram[66] of sites by adherence to three core conditions of EU law. Sites meeting all three in green.

in August 2019,[67] only 20.35% of the top 10K UK websites are reported to use a CMP (from any vendor). 1191 of those (i.e., 58.52%) use the top 5 CMPs, which means the 680 instances our scraper captured represents 57.09% of the total population[68].

We found that implicit consent is common among these sites (32.5%). An array of actions that websites count as consent (but which EU law does not) was extracted from their code, such as just visiting the site (16.8%), navigating within the site (6.2%), revisiting/refreshing the page (7.6%), scrolling or clicking on the page (5.3%) or closing the pop-up or banner (1.6%). 9% of sites accepted more than one form of implicit consent. With only a handful of idiosyncratic exemptions all implied consent was found in the use of 'banner' rather than 'barriers' (a barrier style is in fig. 45). Within those CMPs exhibiting explicit consent, there was a roughly even split between the use of barriers and banners (50.3%/49.7%). Popular CMP implementation wizards still allow their clients to choose implied consent, even when they have already indicated the CMP should check whether the visitor's IP is within the geographical scope of the EU, which should be mutually exclusive. This raises significant questions over adherence with the concept of *data protection by design* in the GDPR.

The vast majority of CMPs make rejecting all tracking substantially more difficult than accepting it. 50.1% of sites did not have a 'reject all' button. Only 12.6% of sites had a 'reject all' button accessible with the same or fewer number of clicks as an 'accept all' button. In practice, this means both were accessible on the first page — an 'accept all' button was never buried in a second layer. 74.3% of reject all buttons were one layer deep, requiring two clicks to press; 0.9% of them were two layers away, requiring at minimum three.

Furthermore, when users went to amend specific consent settings rather than accept everything, they are often faced with pre-ticked boxes of the type specifically forbidden by the GDPR.[69] 56.2% of sites pre-ticked optional vendors or purposes/categories, with 54.1% of sites pre-ticking optional purposes, 32.3%

[67] Adzerk, *Adtech Insights — August 2019 Report*.

[68] It should be noted that Adzerk's methodology counts CMPs by URL endpoints of the Javascript files and we found during development that websites frequently include inactive CMPs' .js files. This means that Adzerk's statistics are likely inflated with double-counting, and that our survey is consequently more representative than the 57.09% would indicate.

[69] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and*

pre-ticking optional categories, and 30.3% pre-ticking both. Our scraper was detecting visual status rather than functional status—we do not know the impact on toggling on or off vendors or categories beyond what the CMP tells the user is happening (Matte et al.'s[70] findings indicate 7.7% of CMPs ignore the consent signal submitted by the user).

Sites relied on a large number of third party trackers, which would take a prohibitively long time for users to inform themselves about clearly. Out of the 85.4% of sites that did list vendors (e.g. third party trackers) within the CMP, there was a median number of 315 vendors (low. quartile 58, upp. quartile 542). Different CMP vendors have different average numbers of vendors, with the highest being QuantCast at 542 (table 8). 75% of sites had over 58 vendors. 76.47% of sites provide some descriptions of their vendors. The mean total length of these descriptions per site is 7985 words: roughly 31.9 minutes of reading for the average 250 words-per-minute reader, not counting interaction time to e.g. unfold collapsed boxes or navigating to and reading specific privacy policies of a vendor.

As discussed, we consider that a site is minimally compliant if it has no optional boxes pre-ticked, if rejection is as easy as acceptance, and if consent is explicit. Only 11.8% of sites met these basic requirements. The interaction between the requirements is shown in Figure 46. This varied significantly by CMP vendor — as shown in Table 8, only Quantcast has a non-negligible number of CMPs that we consider minimally compliant (26.2%), with Crownpeak having zero (that we found). This can largely be explained by the non-existence of implicit consent in QuantCast CMPs and their lower levels of pre-ticked boxes.

14.4.4 *Interim Discussion*

Given that all vendors (with the exception of Crownpeak) have examples in the wild of minimally compliant CMPs, it is unclear whether non-compliance is a practical result of sites configuring it in a non-compliant manner, being encouraged to do so by the CMP vendors or, in some cases, running older CMPs without updating them in light of the more publicised nature of the law.[71] Whatever the practical reasons, 11.8% is an extraordinarily low number for seemingly market-leading CMP vendors, and suggests an urgent role for data protection authorities to take action to ensure only correct configurations are permitted.

The dataset in this study will be available to other researchers, and we welcome further research into, for example, the scraped text content of the CMPs, as the 11.8% in this study is a maximum value that is likely to only decrease on consideration of further aspects of the law which are harder to assess in a formulaic manner.

---

on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1, recital 32.

[70] Matte, Bielova, and Santos, 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.'

[71] Note that the recent judgement from the European Court of Justice clarified that these requirements have been part of EU law since 2012, rather than just since the GDPR (Court of Justice of the European Union, *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:801*)

14.4.5 *Limitations*

Although we manually validated the scraper, we cannot guarantee that there are no false negatives or false positives in our dataset. Because these CMPs are dynamically rendered via JavaScript, determining whether the state of the DOM scraped is the final one is tricky (further complicated by the fact that Scrapy's engine runs on ECMAScript 2015 making tools to deal with asynchronous execution, such as *async/await*, unavailable). We hardcoded a waiting time of 5-15 seconds between loading the site and scraping the content which should be more than sufficient, but there might be exceptions. The CMP might be customised either by the company or the website owner, thwarting the automated way we identify the presence of elements. Legacy implementations, either from various iterations over the years or because the company has been sold multiple times, also introduced branches in the CMP code we might have missed. While we did our best to identify and work around elements of the CMPs designed to obfuscate their function and prevent automation, deliberate changes to data retrieval are often used to foil research for those studying APIs,[72] and such practices seem likely in this domain also to protect against potential automated regulatory scrutiny.

14.5 STUDY 2: DEMONSTRATING THE EFFECTS OF DESIGNS ON ANSWERS

The goal of the second study was to establish if, and to what extent, certain CMP designs affect the consent answer given by users. We were interested in non-compliant designs that are very prevalent, or designs that are not yet described as non-compliant by the applicable regulation. We conducted two field experiments to establish the effects on user behaviour and consent rate of 1) barrier and banner notifications; 2) equal and unequal prominence of accept all and reject all options on the first page; and 3) the level granularity of consent options on the first page (bulk, purposes, vendors).

14.5.1 *Method*

14.5.1.1 *Design*

The study consisted of two counter-balanced experiments, evaluating a total of 8 different interfaces (fig. 47).

Experiment 1 used a [2x2] latin square, within-subjects, repeated measures design. The independent variables were the NOTIFICATION STYLE (*Barrier; Banner*) and BULK CONSENT BUTTONS (*Accept all+Reject all; Accept all*). The primary de-

[72] Axel Bruns (2019). 'After the 'APIcalypse': Social Media Platforms and Their Fight against Critical Scholarly Research.' In: *Information, Communication & Society* 22.11, pp. 1544–1566. DOI: 10.1080/1369118X.2019.1637447. URL: https://doi.org/10.1080/1369118X.2019.1637447; Tania Bucher (2013). 'Objects of Intense Feeling: The Case of the Twitter API : Computational Culture.' In: *Computational Culture: A Journal of Software Studies* 3. URL: http://computationalculture.net/objects-of-intense-feeling-the-case-of-the-twitter-api/ (visited on 06/17/2019).

Figure 47. The 8 interface conditions: (a) Banner / Accept + Reject; (b) Barrier / Accept + Reject; (c) Bulk; (d) Bulk + Purposes; (e) Banner / Accept; (f) Barrier / Accept; (g) Bulk + Vendors; (h) Bulk + Purposes + Vendors.

pendent variable was the CONSENT ANSWER (*Accept all; Reject all; Submit default; Submit personalised*).

Experiment 2 used a [1x4] latin square, within-subjects, repeated measures design. The independent variable was the CONSENT GRANULARITY (*Bulk; Bulk+Purposes; Bulk+Vendors; Bulk+Purposes+Vendors*) on the first page of the notification. The primary dependent variable was the CONSENT ANSWER (*Accept all; Reject all; Submit default; Submit personalised*).

14.5.1.2 *Participants*

A total of 40 participants successfully finished both experiments, with a mean age of 26.1 and standard deviation of 8.6[73]. The majority, 30, had a university degree (17 Bachelor, 12 Master, 1 Doctorate). Seven had some college credit but no degree, and three a highschool diploma. 28 participants were currently studying and 12 were employed full-time. All participants were residing in the United States for the duration of the study, and did not travel to the EU. We selected this sample to prevent the confounding effects of real CMPs, which would have popped-up on top of our injected one if the participants were in the EU and thus in the regulatory scope of the GDPR. Four participants lived in the EU in the past five years, meaning they might already be familiar with pop-ups from the ePrivacy directive. All participants used Google Chrome as their main browser.

Participants were recruited through one of the author's personal network and a university mailing list. They were offered $50 upon completion of the study, and an additional $10 if they successfully recommended others.

[73] Age was reported using brackets of ten years so we are unable to report the exact range; the answers were assumed to be normally distributed to calculate the mean.

14.5.1.3 *Apparatus and Materials*

The materials and apparatus of this study include a pre-study survey, a browser extension, and a post-study survey.

The pre-study survey consisted of 11 questions designed to gather demographic information (age, employment status, highest degree obtained, country of residence), check whether the participants met the study criteria (devices used to browse the web, main browser, travelling to the EU during the study), and acquire their informed consent.

To expose the participants to the different interface designs in a controlled yet ecologically realistic context, we developed a browser extension that injects different pop-ups into any website that participants would visit during their normal daily browsing (available as open-source after publication). The designs of the eight interfaces (i.e., conditions) were inspired by the designs of the top five Consent Management Platforms also used for the scraper study: QuantCast, OneTrust, TrustArc, Cookiebot, and Crownpeak.

All the text, data processing purposes, and vendor names were created by synthesising those commonly used by a random selection of those CMPS in the top 500 Alexa websites in the UK. The data processing purposes are a combination of the options that the five CMPs give to website owners when they create their own pop-up, or the purposes those websites came up with themselves. The vendor names were copied from existing websites, and picked to represent one of four categories: well-known companies (e.g, "Yahoo!"), foreign companies with English names (e.g., "Beijing Interactive Marketing"), foreign companies with non-English names (e.g., "Programatica de publicidad S.L."), and gibberish names (e.g., "s_vi_bikx7Becalgbkjkxx").

The extension used the open-source JavaScript database PouchDB to store the participants' interactions with the interfaces locally, which was synchronised with a CouchDB instance running on OpenStack over an SSL encrypted connection.

The post-study survey consisted of four questions asking the participants to reflect on their general pop-up answering strategy, showed them a visualisation of their actual answers, and asked them to describe how well those answers fit their ideal preferences.

14.5.1.4 *Procedure*

A recruitment email was sent to potential participants asking them to join a study about web-tracker activity in the United States compared to the European Union, and answer the pre-study survey. Once approved, the participants were assigned and emailed a participant number and a link to the Chrome extension on the Chrome Web Store. After installing the extension, a welcome screen automatically appeared asking the participants to fill in their assigned number. This connected the installation to the participant number in the CouchDB database, where each participant was matched to a pre-determined experiment and condition order. Once the extension was successfully activated, a pop-up appeared notifying the participants the experiment had started. To train the participants and homogenise their understanding of the CMPs they received an additional email that

informed them they might sometimes see consent pop-ups (ostensibly when they were shown the European version of a website instead of the US equivalent), explained how those pop-ups worked, and instructed them to answer the pop-ups according to their preferences.

The extension injected a pop-up every fourth url visited – including navigations on the same page, excluding automatic redirects or urls for which an answer was already recorded – to approximate the realistic frequency with which consent pop-ups are currently shown[74]. Each interface condition was repeated four times, requiring the participants to answer sixteen pop-ups per experiment. All interactions with the pop-up were recorded and timestamped: clicking on the elements, toggling purposes or vendors, scrolling the lists, navigating back and forth between the pages, submitting a consent response. Interfaces which were not interacted with were re-appended to the list of conditions and shown again for a maximum of five times, after which it was recorded as "not answered" (similar to a participant clicking or scrolling through the interface without providing a consent response). Once all conditions of the first experiment were answered, the participant progressed to the second experiment.

After completing both experiments, the participants were notified by email that the study was finished, informed that they could uninstall the extension, and asked to complete the post-study survey. The completion time of the experiment ranged from four days to three weeks, depending on how many unique urls the participant visited per day (e.g., some participants mostly visited the same websites, some went on holiday during the experiment, some installed the extension on their secondary device and only used it a couple days per week).

14.5.1.5 *Data analysis*

Although originally 48 participants finished the experiment, we removed eight of them because they mentioned in the survey that their answers were affected by the study (e.g., some participants said they chose "accept all" because they wanted to give more data, despite the instructions they received). To analyse the effects of interface design on consent answers we created a linear regression model with fixed effects; we treated the participants as a factor to account for their (assumed) stable privacy preferences.

14.5.2 *Results*

14.5.2.1 *General interaction patterns*

Of the four possible consent choices – accept all, reject all, submit preferences, no answer – the vast majority of answers submitted by participants was through the bulk options (89.3%), with a skew towards accepting: 55.2% (707) accept all versus 34.1% (437) reject all. Just 9.7% (124) of answers represent the "submit preference" option, and 0.9% (12) were "no answer" (but recorded interactions). Of those 124 "submit" answers, merely 21 – given by 6 participants – were personalised answers, instead of submitting the default status (all toggled off). Four

[74] Based on Adzerk's Ad-Tech Insights report: (Adzerk, *Adtech Insights — August 2019 Report*)

of those 21 were personalised by clicking the "Toggle All" button, which means *only 17 answers out of 1280 (1.3%) represent a participant consenting to a specific selection of purposes or vendors.* Whether this is because users are unable to make such decisions, are not interested in that level of detail, or fatigued by the form and frequency of the question is unclear; but it does indicate that users' consent is rarely empirically as "specific" as the GDPR requires it to be. It does not follow that specific controls should therefore be removed, but rather that such specificity could be distributed to other actors invited by the user (e.g., browser agents, consent predictors, a knowledgeable friend).

Almost all interactions (93.1%) were limited to the first page of the pop-up the participants were exposed to. Seven out of eight interfaces had a "more options" link to navigate to a second or third page for more information and granular consent choices, but this was clicked only 88 times (6.9%). When participants were exposed to a scrollable list of data collection purposes or vendors on the first or subsequent pages (560 occasions), they ignored it 68.6% (384) of the time. Of the 176 instances they did scroll, 21.6% (38) were between 0 and 25 percent of the list, and 64.2% (113) between 75 and 100 percent. In other words, anything not immediately visible to the user, anything requiring interaction to access, might as well not exist.

14.5.2.2 *Notification style*

The validity of one design element still under discussion by policy makers is that of the notification style:[75] a barrier in the middle of the screen which prevents the user from interacting with the website until a response is recorded, or a banner stretching the width of the screen that does not block access to the information .

We found that *notification style did not affect the consent rate of participants*. Two simple linear regressions were calculated to investigate the relationship between the answer given (accept or not) and notification style (banner or barrier). The first, comparing BARRIER to BANNER with both the Accept and Reject button, did not find a regression line at all ($F(1,279) = 0.000$, $p = 1$). The second, comparing BARRIER to BANNER with just the Accept button, found a non-significant relationship ($p = 0.702$), with a slope coefficient of 0.013 (95% CI min and max of -0.052 and 0.077 respectively) and an R^2 of 0.001.

While there was no difference in acceptance rate when participants actually answered the pop-up, the banner notification was ignored 3.6 times more often than the barrier. For this statistic we considered any pop-up that was not interacted with, but which had a time difference of at least 3 seconds between being injected and the tab being closed, as "ignored"; 133 of such instances were found, with only 21.1% (28) for the barrier and 78.9% (106) for the banner.

14.5.2.3 *Button prominence*

Data from our scraper indicates 'accept all' and 'reject all' buttons are not displayed with equal prominence: only a mere 12.6% of sites show both on the

[75] Zuiderveen Borgesius et al., 'Tracking Walls, Take-It-Or-Leave-It Choices, the GDPR, and the ePrivacy Regulation.'

same page. Such unequal prominence of consent options is already considered non-compliant with the GDPR[76] because it is expected they affect consent answers, but the severity of its impact is unknown.

We found that *removing the 'reject all' button from the first page increased the probability of consent by 22–23 percentage points*. We calculated two simple linear regressions to analyse the relationship between the answer given (accept or not) and the consent options on the first page (accept and reject, or just accept). The first, comparing ACCEPT ALL + REJECT ALL to ACCEPT ALL for the barrier notification, found a strong positive linear relationship between the two. The significant ($p < 0.001$) slope coefficient for the consent answer was 0.220, meaning the accept rate increased on average by 22.0 percentage points when the reject all button was removed from the first page. The 95% CI had a minimum and maximum of 0.149 and 0.290 respectively. The R^2 was 0.117, so 11.7% of the variation in answers for the barrier notification can be explained by the changing prominence of the buttons.

The second regression compared ACCEPT ALL + REJECT ALL to ACCEPT ALL for banner notifications and found a similarly strong, positive linear relationship between the button prominence and answer given. The significant ($p < 0.001$) slope coefficient was 0.231, meaning the accept rate increased on average by 23.1 percentage points when the reject all button was removed from the first page. The 95% CI had a minimum and maximum of 0.163 and 0.230 respectively. The R^2 was 0.135, so 13.5% of the variation in answers for the banner notification can be explained by the changing prominence of the buttons.

14.5.2.4 *Level of granularity*

The most common order in which consent options are displayed is bulk first, followed by the data collection purposes on the second page and the vendors on the third page, or some combination of those two on the same page.

We found that *displaying more granular consent choices on the first page decreased the probability of consent by 8–20 percentage points*. We calculated a simple linear regression to compare a BULK only interface to an interface that combined BULK + PURPOSES; BULK + VENDORS; and BULK + PURPOSES + VENDORS on the same page. We found a significant ($p < 0.01$) negative relationship between all increases in the level of granularity of consent options and the answer given, with different strengths depending on the kind of options that were available. As illustrated by Table 9, showing just the vendors affected the acceptance rate the most (−0.200), whereas just the purposes (−0.088) and the combination of vendors and purposes (−0.119) were closer together but still lower than the baseline interface with just bulk options. Along the same lines, the 95% CIs overlap most between PURPOSES and PURPOSES + VENDORS and only a little with VENDORS.

[76] Advocate General Szupunar, *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:246, Opinion of the Advocate General*; Information Commissioner's Office, *Guidance on the Use of Cookies and Similar Technologies*.

	Dependent variable:	*95% CI:*
	'accept all' clicked	lower : upper
Bulk + Purposes	−0.088**	−0.151 : −0.024
	(0.032)	
Bulk + Vendors	−0.200***	−0.263 : −0.137
	(0.032)	
Bulk + Purposes	−0.119***	−0.182 : −0.056
+ Vendors	(0.032)	
Observations	640	
R^2	0.062	
F Statistic	13.210*** (df = 3; 597)	
Note:		*p<0.1; **p<0.01; ***p<0.001

Table 9. Level of granularity on the first page, with bulk consent as the reference

14.5.2.5 *Participant Strategies and Behaviour Patterns*

While the experimental data suggests how different designs affect how "freely given" the consent answers of participants are, it does not provide information about how those answer relate to their preferred privacy settings. In a post-study survey, we requested participants to describe their overall answering strategy, showed them a visualisation of their actual behaviour, and then asked them to state how well their answers reflected their ideal preferences and, if not, why. To structure these findings, we classify participants according to their general consent answers: always accept, mostly accept (>= 75%), mixed consent, mostly reject (>= 75%), always reject.

When asked what they based their choices on, the answers touched on eleven different topics. The four 'always accept' participants cited a general apathy towards privacy concerns and "*just did it to make the window go away*". The one participant that 'always rejected', no matter whether that required more effort, argued that they would only accept data collection if it was to use a particular feature offered by the site. The eleven participants categorised as 'mostly reject' heavily emphasised a disagreement with the practice of tracking in general and stated they would only consent to have their data collected if it was for websites they trusted. Two of those also mentioned that they did not feel a need for any personalisation. The participants that fell into the 'mostly accept' and 'mixed consent' category were more diverse. Most often mentioned were pragmatic reasons such as just wanting to get to the site as quickly as possible, not believing the controls were meaningful, and not wanting to lose any functionality. Eight decided based on trust, whether it was the website or the vendors, and the sensitivity of the data they would be submitting (e.g., banking information). One participant stated that they relied on other methods to protect their privacy, so did not

care that much about their pop-up answers: "*I tend to vary my devices/browsers/accounts/use incognito and duckduckgo a lot, I'm not too worried about my data being tracked to every detail*."

After being shown a visualisation of their actual consent behaviour and asked if it matched their ideal settings, the responses were predominately that it did not. Only those falling into the two extreme categories – 'always accept' and 'always reject' – all indicated they agreed (3) or strongly agreed (2) with their answers. For the remaining three categories, the sentiments were mostly spread evenly along the spectrum, with 11 somewhat agreeing, 3 neither agreeing nor disagreeing, 7 somewhat disagreeing, 1 disagreeing, and 3 strongly disagreeing.

The 25 participants who indicated their behaviour did not match their ideal privacy settings were asked to explain what the reason for this difference was. Participants mentioned desires such as just wanting more privacy ("*I would rather companies not collect any information*"); the fear of unknown consequences of opting-out ("*I didn't want to risk the website not working after that*"); and not knowing what their ideal preferences even are. The most common reason mentioned, however, was the interface design. Participants lamented the fact that pop-ups stand in the way of their primary goal (accessing a service), that the frequency of the pop-ups caused frustration and consent fatigue, and even the perception that the pop-up "*forced them to accept*" – even though these options were available on the second page.

14.5.3 *Interim Discussion*

The experimental results indicate how two of the most common consent interface designs – not showing a 'reject all' button on the first page; and showing bulk options before showing granular control – make it more likely for users to provide consent, violating the principle of "freely given"[77]. The notification style, on the other hand, appears to have no effect on the answer, but possibly a large effect on whether an answer is given at all, suggesting that a non-blocking mechanism provides a desired third consent option to users: a neutral middle-ground. The qualitative reflections of the participants, however, put into question the entire notice-and-consent model not because of specific design decisions but merely because an action is required before the user can accomplish their main task and because they appear too frequently if they are shown on a website-by-website basis.

14.5.4 *Limitations*

The participant sample is by no means representative of the general population in the United States: they are almost all young and university-educated, and recruited primarily through an emailing list of a computer science department. Arguably, this means that our results describe a "best case scenario": these participants should be more knowledgeable about privacy issues and better equipped to understand consent interfaces than the average web user.

[77] It should be noted this data alone is not enough to establish legal compliance.

There are a number of confounding variables that could have affected the participants' answers. First, although the condition order was counterbalanced, we cannot guarantee that the participants were actually exposed to them in that order (e.g., if they opened multiple tabs in a row and visited them anachronistically), meaning order effects might not be controlled for. Second, because we showed the same pop-up to each participant until we recorded four answers per interface, some participants were exposed to the different conditions more often than others. Lastly, participants might have also encountered "real" pop-ups at the same time as the injected ones if the website they were visiting was within the territorial scope of the GDPR.

While the GDPR is a European policy, our experiments were conducted in the United States. These populations have been exposed to different legal regimes and different consent controls over the year, something which we expect has affected their mental model of these kind of pop-ups and accordingly, how they answer them. This might influence the extent to which these findings can be generalised to a European population, and thus how they should be used to inform EU policy changes.

14.6 DISCUSSION AND CONCLUSION

The results of our empirical survey of CMPs today illustrates the extent to which illegal practices prevail, with vendors of CMPs turning a blind eye to — or worse, incentivising —- clearly illegal configurations of their systems. Enforcement in this area is sorely lacking. Data protection authorities should make use of automated tools like the one we have designed to expedite discovery and enforcement. Designers might help here to design tools for regulators, rather than just for users or for websites. Regulators should also work further upstream and consider placing requirements on the vendors of CMPs to only allow compliant designs to be placed on the market. Such enforcement may be possible as the Court of Justice indicates that plugin system designers can be 'joint controllers' along with websites,[78] and the UK's ICO indicates it may be willing to force advertising trade bodies to alter their standards.[79] If this is the case, regulators must carefully consider how to build a robust and well-maintained evidence base for user-centric CMP design.

A core takeaway from the user study is that placing controls or information below the first layer renders it effectively ignored. This leaves a few options for genuine control of tracking online. If the notice-and-consent model is to continue, it may be necessary to declare that, for example, consent can never be valid with the presence of the (on average) hundreds of third parties we have shown data is

[78] Court of Justice of the European Union, *Case C-49/17* Fashion ID GmbH & Co.KG v Verbraucherzentrale NRW eV. *ECLI:EU:C:2019:629*; Rene Mahieu, Joris van Hoboken, and Hadi Asghari (2019). 'Responsibility for Data Protection in a Networked World: On the Question of the Controller, Effective and Complete Protection and Its Application to Data Access Rights in Europe.' In: *Journal of Intellectual Property, Information Technology and Electronic Commerce Law* 10.1, pp. 84–104. (Visited on 09/03/2019); Brendan Van Alsenoy (2019). *Data Protection Law in the EU: Roles, Responsibilities and Liability*. Cambridge: Intersentia.

[79] Information Commissioner's Office, *Update Report into Adtech and Real Time Bidding*.

sent to and cookies laid by today. This would mean consent would only be valid if a compact but representative and rich description can be placed on the first layer, and could certainly be a possible direction for the Court of Justice to consider if they interpret the principles of data protection in a future case.

An alternative approach would be to overhaul the design pattern of the consent banner or barrier, and have richer, more durable ways to set preferences, potentially within the browser. The key is that such browser settings would be legally binding, rather than weak and self-regulatory in nature. Yet the current heavy lobbying around the EU's draft ePrivacy Regulation has centred in part on adtech firms trying to prevent browser settings having legally binding effect — part of an ongoing drama for many years about the potential legal status of 'Do Not Track' signals.[80] Designers have a role here: how can users reflect on tracking across the Web, rather than on a per-site basis? If users are not to automatically reject everything, how can advertisers negotiate and present them with reasons that they should consent? Might there be a role for delegation of preferences to a trusted civil society actor, and what kind of relationship, information and interaction might the user have with these? We invite and encourage researchers to bring their skills and views to bear on these important, current issues at the confluence of regulation, design and fundamental rights.

[80] Irene Kamara and Eleni Kosta (2016). 'Do Not Track Initiatives: Regaining the Lost User Control.' In: *International Data Privacy Law* 6.4, pp. 276–290. ISSN: 2044-3994. DOI: 10/gdxwds. (Visited on 12/28/2018).

15.

NEGOTIATING CONSENT POP-UPS WITH SUPERVISORY AUTHORITIES IN DENMARK

15.1 INTRODUCTION

The majority of consent pop-ups on the web do not meet the requirements for legally valid consent laid out in the General Data Protection Regulation (GDPR),[1] undermining the regulation and the rule of law. This begs the question: How can we make a meaningful, lasting change to this status quo? Although the battle around data processing and privacy stretches back decades (see Chapter 13), the adoption of the GDPR has brought data protection laws back into mainstream focus, and the extensive promotion of the regulation has committed the EU's legitimacy to its success. These changes have created momentum that might shake up the tiresome tolerance of illegal web-tracking.

It is unclear how to best leverage that momentum to address the lack of compliance in consent pop-ups. The GDPR and ePrivacy Directive have given legal responsibility over the design of consent notices to four parties: the *data subject*, the *data controller(s)*, the *data processor(s)*, and the *supervisory authority*. This chapter examines the following theory: a) the dominant designs of consent pop-ups can be changed through a negotiation between a *data subject* and a *supervisory authority*; and b) software can be used as an effective diagnostic tool[2] to facilitate that negotiation.

First, I will introduce the relevant supervisory bodies and the laws from which they derive their authority. Second, I will describe the enforcement status quo, with a specific focus on the Danish context. Third, I will recount the negotiation process with the authorities through the use of *automated compliance monitoring software*. Last, I will show how this has changed the design of consent pop-ups on the Danish web, where the proportion of pop-ups with a reject button on the first page has increased from 1,5% to 41%.

[1] Nouwens et al., 'Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence'; Célestin Matte, Nataliia Bielova, and Cristiana Santos (2019b). 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.' In: *Under review*. URL: arXiv:1911.09964.

[2] Abebe et al., 'Roles for computing in social change.'

15.2 SUPERVISORY AUTHORITIES

In the context of web-tracking technologies, there are two legislative documents from which supervisory authorities derive their power: the General Data Protection Regulation, and the national transposition of the ePrivacy Directive. The bodies responsible for the enforcement of the GDPR are the Data Protection Authorities, and for the ePrivacy Directive, whichever authority the Member State has elected to appoint for the task. These are the organisations that oversee the design of consent pop-ups and who have a duty to respond to individuals exercising their data protection rights.

15.2.1 *The ePrivacy Directive enforcement authority*

The original ePrivacy Directive adopted in 2002 does not specify how the law should be enforced, who should be in charge, and what powers they have. The requirement of an enforcement agency is (strangely) first introduced as part of Article 15a of the 2009 amendment, which states that each EU Member should appoint and provide resources for a "competent national authority" with the power to start investigations and intervene when there are transgressions of the law.[3] Because it is a Directive and thus does not strive for maximum harmonisation across the EU, each Member State was allowed to make their own decision about who that authority would be. Generally, most governments elected to give the responsibility to (a number of) already existing institutions, such as telecomunnications ombudsmen, consumer protection boards, electronic communication and postal regulators, competition authorities, or personal data protection agencies.

In Denmark, the ePrivacy Directive was transposed into a total of fourteen different documents, the most relevant ones for web-tracking and consent being the *Electronic Communications Networks and Services Act*[4] and the *Executive Order on Information and Consent Required in Case of Storing and Accessing Information in End-User Terminal Equipment*.[5] The Act originally appointed the *IT- og Telestyrelsen* (IT and Telecom Agency) as the main responsible authority in 2002, but this agency was abolished following the 2011 parliamentary elections and its responsibilities were redistributed to the Ministry of Industry, Business, and Financial Affairs; Ministry of Defense; Ministry of Finance; and Ministry of Economic Affairs and the Interior (some of which have since been disbanded and/or merged themselves). Since 2011, the main responsible body for cookies and consent has been the *Erhvervsstyrelsen* – the Danish Business Authority – which "works to make it easy and attractive to run responsible businesses in

[3] European Union (2009). 'Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services.' In: *Official Journal of the European Union*, Article15a.

[4] Energi-, Forsynings- og Klimaministeriet (2011). *Lov om elektroniske kommunikationsnet og -tjenester. LOV nr 169 af 03/03/2011*. URL: https://www.retsinformation.dk/eli/lta/2011/169.

[5] Erhvervsministeriet (2011). *Bekendtgørelse om krav til information og samtykke ved lagring af eller adgang til oplysninger i slutbrugeres terminaludstyr*. URL: https://www.retsinformation.dk/eli/lta/2011/1148.

Denmark".[6] It has a broad portfolio that includes all applicable laws which determine the responsibilities and rights of businesses when operating in Denmark (e.g., environmental guidelines, trade laws, development funds). For some parts of the Directive, the responsibility of upholding it is delegated to *Teleklagenævnet* (the Telecommunications Complaints Board), *Forbrugerombudsmanen* (the Danish Consumer Ombudsman), and *Konkurrence- og Forbrugerstyrelsen* (the Danish Competition and Consumer Authority).

15.2.2 *The GDPR enforcement authority*

Data Protection Authorities (DPAs) in the European Union are independent organisations tasked with monitoring the application of data protection law. Under the GDPR, each Member State is required to have their own national DPA and is responsible for providing it with the human, technical, and financial resources necessary to carry out its tasks and exercise its investigative, corrective, and advisory powers.[7] They provide guidance about how to comply with the law, deal with complaints, carry out investigations, and function as the central contact point about the rights and responsibilities of different actors vis-à-vis data protection law. They do not have any judicial powers, but they are able to issue fines for transgressions of up to €20 million or 4% of the total annual turn-over. The responsibilities of the DPAs are mostly within the national borders they operate in, although they are expected to cooperate with the other DPAs across the EU. If a case is launched against an organisation that operates within multiple EU countries, the responsibility will lie with the DPA in the country where that organisation has its main establishment. This regulatory mechanism, called the one-stop-shop, was introduced as part of the GDPR to make it easier to deal with cross-border cases and ensure consistent application across the Union.

In Denmark, *Datatilsynet* is the Data Protection Authority that takes the lead on anything concerning data protection law. However, because of the overlap between the GDPR and ePrivacy Directive regarding cookies and consent, there is some ambiguity around who is responsible for enforcement and should provide guidance. *Datatilsynet* has stated in the media that *Erhvervsstyrelsen* should be the primary authority, but they have also recently started investigations and released verdicts about illegal consent pop-ups themselves. They started to provide their own guidance around pop-up design, which in certain places actually contradicts that of *Erhvervsstyrelsen*.

[6] Erhvervsstyrelsen (2020). *Om Erhvervsstyrelsen*. URL: https://erhvervsstyrelsen.dk/om-erhvervsstyrelsen (visited on 07/06/2020).

[7] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*, Article52(4.

15.3 ENFORCEMENT STATUS QUO

15.3.1 *ePrivacy Directive*

There has been little enforcement of the ePrivacy Directive since the latest amendment in 2009 (which was only successfully transposed by all Member States in 2013, two years after the official deadline). In a 2015 assessment of the Directive's effectiveness for the European Commission, the authors euphemistically describe that "[i]t is ... difficult to deny that the introduction of the consent rule in Art. 5.3 did not entirely reach its objective".[8] Most authorities have not provided more enforcement beyond writing documents explaining the law and how to comply with it, but even those, the report describes, are not always correct. Also in 2015, the European Commission announced that they were working on replacing the Directive with the ePrivacy Regulation (ePR)[9] to address its shortcomings and complement the GDPR. Although it was supposed to be adopted concurrently with the GDPR, it has stalled significantly due to heavy lobbying and it is currently unclear when or whether it will materialise.

There are no clear records about how many cases *Erhvervsstyrelsen* has handled since 2011 that address consent pop-ups; in response to a media inquiry they stated that "they do not have an estimate of the number of inquiries specifically regarding cookies".[10]

15.3.2 *General Data Protection Regulation*

Two years after the GDPR has come into effect, it is increasingly criticised for a lack of enforcement. Academic studies and investigative journalists have shown the many areas in which commercial companies are failing to comply, yet are not being held accountable. One of the main reasons suggested for this is the lack of resources and staff provided to the DPA, something which Member States are legally obliged to do under Article 52(4).[11] The European Data Protection Board – the umbrella organisation made up of the head of each national DPA – released a study in February 2019 reporting that 94% of DPAs did not have

[8] European Commission, Directorate-General for the Information Society and Media (2015). *ePrivacy directive, assessment of transposition, effectiveness and copmatibility with the proposed data protection regulation final report.* URL: http://bookshop.europa.eu/uri?target=EUB:NOTICE:KK0415268:EN:HTML.

[9] European Commission (2017). *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Communications)*.

[10] Emilie Aagaard (2020). 'Forskere: DR's og Folketingets hjemmesider er på kant med persondataloven.' In: *Danmarks Radio*.

[11] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*.

sufficient resources to carry out their tasks.[12] Although the GDPR aimed to unify and equalise the regulation and its enforcement throughout the EU, a recent report by the Brave Software – developers of an open-source browser that blocks ads and trackers – shows there are considerable differences between the different national DPAs. Half of the DPAs receive €5 million or less budget per year (fig. 48) and only six of twenty eight DPAs have more than 10 tech specialist employees (fig. 49).

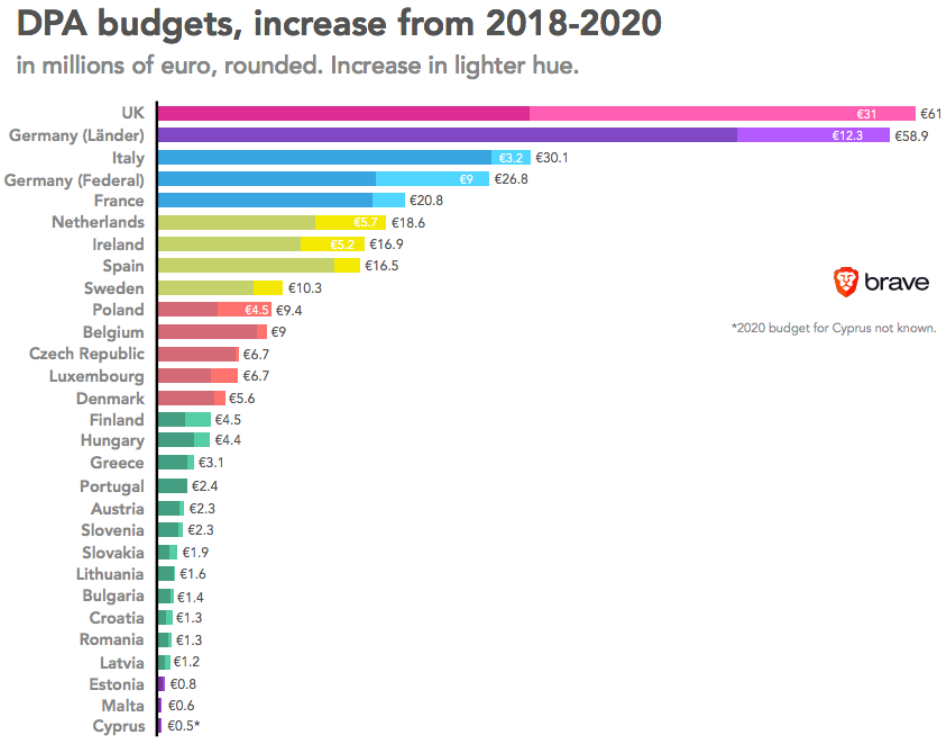


Figure 48. Annual budgets of the DPAs, via Brave[13]

Another main concern is the efficacy of the one-stop-shop system. Luxembourg, the Netherlands, and Ireland disproportionally house large multinational companies – e.g., Facebook, Google, Amazon – because of their lucrative tax schemes. As a result, any complaints brought against these companies in other Member Countries have become the responsibility of, mostly, the Irish DPA, which has been unable to cope and considerably stalled any actions against transgressions of the largest global players.

In Denmark, *Datatilsynet* has since May 2018 made only one verdict related to consent pop-ups. The original complaint was made by Christian Schmidt, a private individual, on August 29, 2018 against dmi.dk, the official website of the Danish Meteorological Institute. It took until February 11, 2020 until the verdict was released that DMI's consent pop-up was non-compliant.

[12] European Data Protection Board (2019). 'First overview on the implementation of the GDPR and the roles and means of the national supervisory authorities.' In: URL: https://edpb.europa.eu/sites/edpb/files/files/file1/19_2019_edpb_written_report_to_libe_en.pdf.

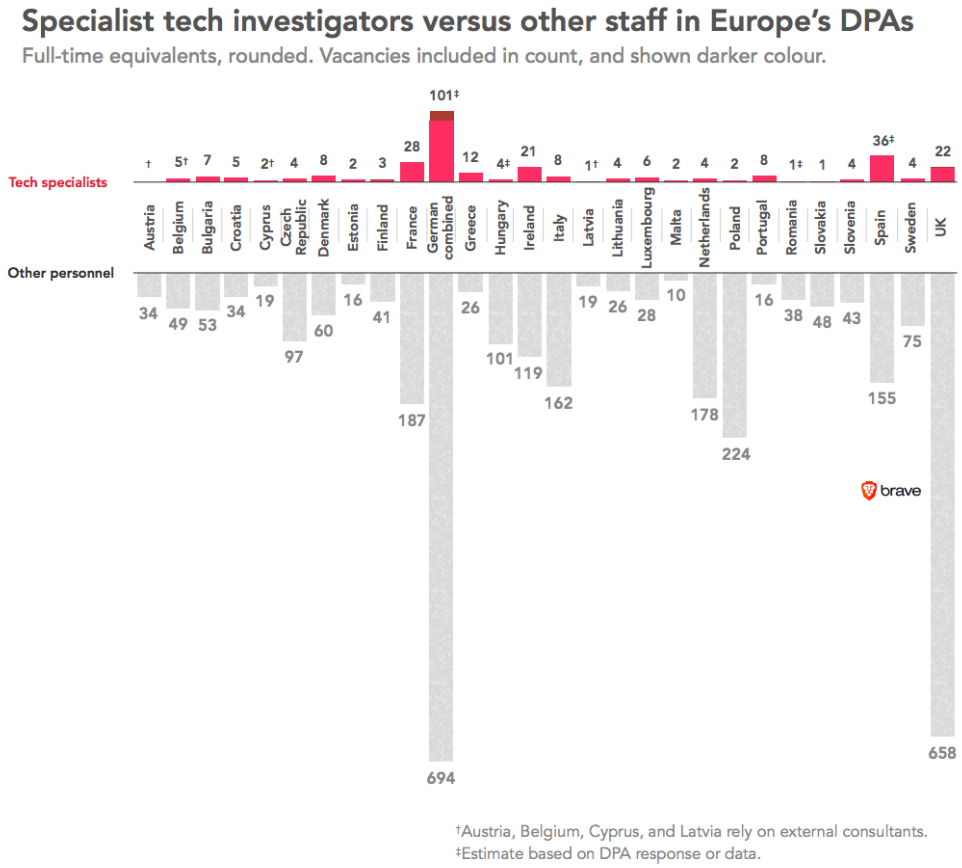


Figure 49. Number of tech specialists and total employees per DPA, via Brave[14]

15.4 THE NEGOTIATION PROCESS

Given the overall lethargic enforcement of the GDPR, combined with the lack of resources of the DPAs and thus their limited bandwidth, negotiating the design of consent pop-ups with/through them required two things: gathering the data to demonstrate the lack of compliance, and making them prioritise this problem.

15.4.1 *Gathering the Data*

DPAs generally use small-scale, qualitative methods when investigating the non-compliance of an industry (see e.g., the Irish DPA's 2019 report[15]) for two reasons: 1) some of the data protection law's requirements can only be assessed qualitatively; and 2) they rarely have the technical expertise or resources to use computational methods. As a result, they do not have a large-scale overview of the consent pop-up situation within their borders.

[15] Data Protection Commission (2020). 'Report by the Data Protection Commission on the use of cookies and other tracking technologies: Following a sweep conducted between August 2019 and December 2019.' In:

As part of the study described in Chapter 14 I built an *automated compliance monitoring software* (a web-scraper) which analyses the designs of six of the most popular third-party consent pop-ups: Quantcast, OneTrust, TrustArc, CookieBot, Cookieinformation, and Crownpeak. It takes a plain-text list of urls as input and returns a new-line delimited JSON file for each domain. Table **??** lists all variables collected and Figure 50 shows an example response.


```
{
    cmp: "cookiebot",
    notificationStyle: "barrier",
    bulkDescription: "Samtykke til DR's brug af cookies \n\n DR indsamler oplysninger om dine
    acceptAll: true,
        rejectAll: null,
        consentAction: {
        visitPage: false,
            scrollPage: true,
            navigatePage: false,
            closePopup: false,
            refreshPage: false,
            clickConsentButton: true
    },
    acceptAll: {present: true, label: "Tillad alle cookies", clicks: 0},
    rejectAll: {present: true, label: "Tillad udvalgte", clicks: 0},
    purpose: {
        present: true,
        clicks: 0,
        labels: [
            {name: "Nødvendig",
                description: "Nødvendige cookies er tekniske cookies, der sørger for, at dr.d
                defaultStatus: true,
                enabled: false},
            {name: "Præferencer"...},
            {name: "Statistik"...},
            {name: "Markedsføring"...}
        ]
    },
    vendor: {
        present: false,
        clicks: null,
        labels: [
            {name: "demdex",
                description: "Adobe Inc. Indsamler anonym statistik om brugerens besøg på hje
                defaultStatus: true,
                enabled: false},
            {name: "_ip4_u"...},
            {name: "everest_session_v2"...},
            {name: "_twitter_sess"...},
            {name: "personalization_id"...},
        ]
}
```


Figure 50. Example of the JSON object returned by the scraper for a given domain

15.4.2 *Raising the Priority*

DPAs operate with limited resources and thus have to prioritise their tasks. The only formal way to influence the DPAs priorities is by launching a complaint, and while they are required to respond within three months, there is no time limit on reaching an actual verdict. Rather than rely on the bureaucratic process, I used the acceptance notification of the research paper described in Chapter 14

Date	Publisher	Title
10/01/2020	TechCrunch	Cookie consent tools are being used to undermine EU privacy rules, study suggests
13/01/2020	Engadget	Most websites don't follow European cookie consent laws, study shows
13/01/2020	Vice	Companies Use 'Dark Patterns' to Mislead Users About Privacy Law, Study Shows
17/01/2020	de Standaard	Websites schenden massaal privacywetgeving
18/01/2020	Danmarks Radio	9 ud af 10 hjemmesider overholder ikke lov om cookies: 'Det kan virkelig give bagslag'
21/01/2020	le Figaro	RGPD: neuf sites sur dix rusent pour obtenir le consentement des internautes
22/01/2020	Danmarks Radio	Forskere: DR's og Folketingets hjemmesider er på kant med persondataloven
22/01/2020	Danmarks Radio	Accepterer du cookies? Her er fire metoder til at undgå overvågning på nettet
08/02/2020	BBC	Cookies crumbling as Google phases them out
11/02/2020	Fast Company	Why you can't escape dark patterns
29/03/2020	Prosa	Virksomheder narrer brugerne til mere dataovervågning
28/05/2020	Wired	We need to fix GDPR's biggest failure: broken cookie notices

Table 11. News reporting on data collected about illegal consent pop-ups

to contact the press. The overwhelming illegality of consent pop-ups, contrasted with the bombastic release of the GDPR a year and a half before, proved interesting enough that various international media sources published articles about the issue (table 11). The public Danish Broadcasting Corporation (*Danmarks Radio*) released a three part series on the data, one discussing the situation in general and holding the Danish supervisory bodies to account, one highlighting how their own site, the Danish parliament, and other public institutions were non-compliant, and one educating readers about ways to protect themselves against web-tracking.

The media attention had the desired effect, and resulted in the study being discussed by the Irish Data Protection Commissioner Helen Dixon at the annual Computers, Privacy, and Data Protection conference and a meeting with the Danish DPA; as well as an invitation from the company behind the privacy-preserving Brave browser to present at their London office; meetings with leading Danish consent management company Cookieinformation; and various invitations for conference panels.

15.5 NEGOTIATION OUTCOME

On the 17th of February, just a day short of one month since the first DR articles came out criticising Datatilsynet and Erhvervsstyrelsen, the DPA released new guidance for the design of consent pop-ups and reached a verdict in a case against the Danish Meterological Institute's pop-up that had been ongoing since

August 2018.[16] Politiken, one of the leading broadsheet newspapers, reported on the case with the headline "A bomb under websites: DMI receives severe criticism in groundbreaking decision".[17] Ehrvervsstyrelsen, although adament in their original comments to DR in February that their guidance was up-to-date and correct, also changed their cookie guidance page.

The verdict explained how the design of DMI's pop-up did not result in legally valid consent, and emphasised that 1) it should be as easy to refuse consent as it is to give it – a "one-click-away" approach or differently styled buttons are not sufficiently transparent; and 2) it should be easy to access and understand which data controllers a visitor is giving consent to – using websites, nicknames, or product names is not enough.

The verdict and guidance has had considerable effect on the design of consent pop-ups on the Danish web. I used the *automated compliance monitoring software* to analyse the designs of consent pop-ups on the top 30.000 most popular websites in Denmark on a weekly basis since February 18. The data shows that the presence of a reject button on the first page – one of the main points raised in *Datatilsynet's* verdict – has steadily increased from 1,5% to 41% (fig. 51). However, it is striking how almost all of that increase can be attributed to just a single third-party consent pop-up provider: Cookieinformation. The proportion for the only other company with a non-negligable market share – CookieBot – has stayed almost the same (from 4,4% to 5,7%). Despite the positive change, still not even half of all pop-ups make it as easy to reject consent as to give it.

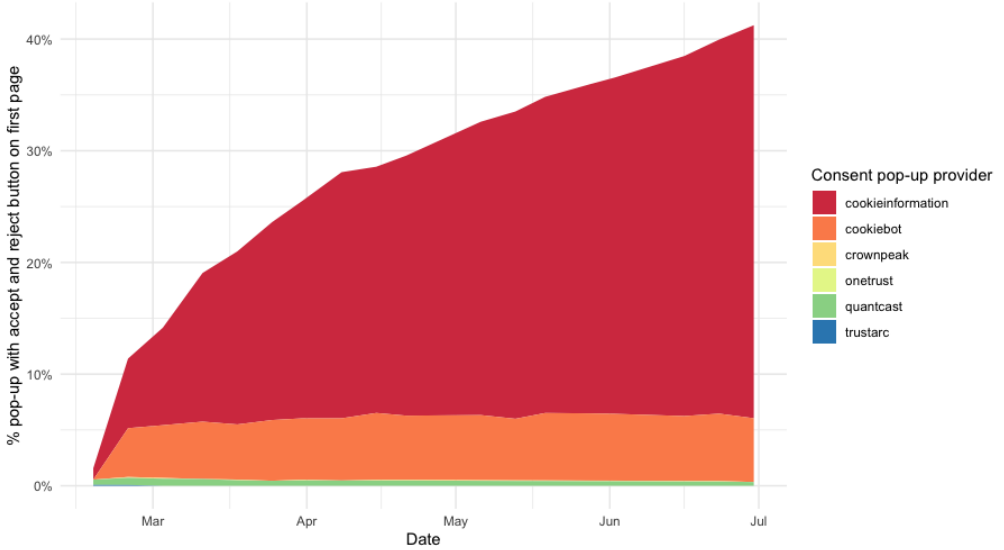


Figure 51. Percentage of pop-ups with a reject button on the first page on the top 30.000 most popular Danish websites

[16] Datatilsynet (2020). *DMI's behandling af personoplysninger om hjemmesidebesøgende*. URL: https://www.datatilsynet.dk/tilsyn-og-afgoerelser/afgoerelser/2020/feb/dmis-behandling-af-personoplysninger-om-hjemmesidebesoegende.

[17] Jakob Sorgenfri Kjær (2020). *Bombe under hjemmesider: DMI får alvorlig kritik i banebrydende afgørelse*. URL: https://politiken.dk/viden/Tech/art7657203/DMI-f\%C3\%A5r-alvorlig-kritik-i-banebrydende-afg\%C3\%B8relse.

In addition to the relative presence of reject buttons, the number of consent pop-ups has also increased by more than 60% (fig. 52). However, the total percentage of sites using one of these six third-party services is still less than 10%.

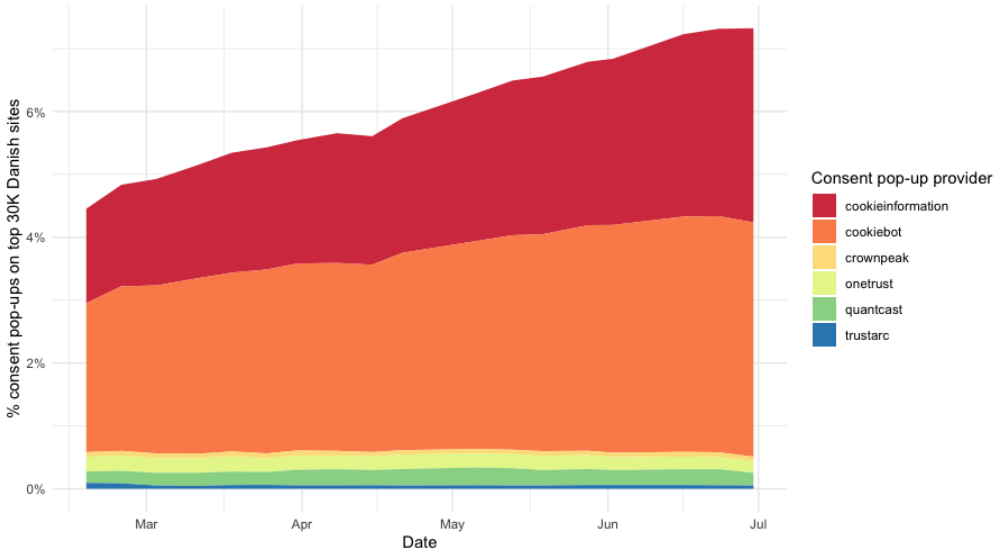


Figure 52. Percentage of sites using one of six third-party consent pop-up providers on the top 30.000 most popular Danish websites

15.6 CONCLUSION & FUTURE WORK

The design of consent software can be changed, despite decades of stagnation, by negotiating with supervisory authorities; and software can be an effective diagnostic tool during this process.

Although this negotiating process has achieved considerable success, it is unclear whether supervisory authorities are the most effective target to affect change. The data shows how just one consent pop-up provider was almost single-handedly responsible for all the increase in reject button compliance. This strongly suggests that focusing on these data processors(/joint controllers) would be more efficient than intermediary DPAs. Future work should analyse what caused the two major Danish consent providers – Cookieinformation and CookieBot – to have such starkly different trajectories in their compliance rates after the *Datatilsynet* verdict. Did Cookieinformation remove non-compliant designs from their portfolio? Did they more aggressively inform their clients? Is there a self-selection bias in types of clients between the two companies?

If consent providers are the key to more effective enforcement, Data Protection Authorities might explore requiring their software to be inherently interoperable, which would allow automated tools to more easily (and accurately) analyse their designs (rather than the more fragile and time-consuming reverse-engineering approach used by my software). Such requirements could render the industry more legible and thus governable. This recommendation comes with a strong caveat: as data-driven governance and evidence-based policy approaches continue to grow in popularity, we should stay vigilant that our institutions do

not fall victim to techno-solutionism. While enforcement in data protection law is sorely lacking, the successful use of computational tools should not be allowed to direct undue attention and resources to only those aspects of data protection and privacy that can be monitored and operationalised through such tools. Already it was clear that part of the interest and perceived legitimacy of my study came from the quantity of the data; much reporting focused on the memetic fact that 10.000 websites were scraped. While it worked in my benefit, and was hugely tempting to leverage, future use of *negotiation software* should more clearly identify what aspects of the law and reality are less amenable to computerisation and datafication.

16.

NEGOTIATING CONSENT POP-UP DESIGNS USING ADVERSARIAL INTEROPERABILITY

In collaboration with: Rolf Bagge, Janus Bager Kristensen, and Clemens Nylandsted Klokmose

16.1 INTRODUCTION

The majority of consent pop-ups on the web do not meet the requirements for legally valid consent laid out in the General Data Protection Regulation (GDPR),[1] undermining the regulation and the rule of law. Supporting the supervisory activities of DPAs as described in Chapter 15 is an obvious first step towards improving compliance, but this path to software change is dependent on the DPAs ability and willigness to enforce the regulation, a necessarily slow and procedural process. A direct action approach, instead, would allow a data subject to negotiate the design of software immediately, and accomplish their goals independent of other stakeholders.

This chapter explores how to negotiate the interaction design of consent pop-ups directly by interoperating with the code of the software. This would allow data subjects to bypass the non-compliant designs of consent pop-ups and enforce their data protection preferences, without relying on the cooperation of external parties.

First, I will introduce existing approaches to software-mediated data protection negotiation, and the concept of interoperability in HCI research. Second, I will describe the technical implementation of the five most popular third-party consent management platforms (CMPs), and how their designs support or inhibit interoperability. Third, I will present *consent-automating software*: a browser extension which automatically answers consent pop-ups using *adversarial interoperability* – the ability to interact with systems in ways that go against their fundamental interests. Last, I will reflect on the impact of this negotiation software, and what

[1] Nouwens et al., 'Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence'; Matte, Bielova, and Santos, 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.'

policy changes would improve reliable interopability and make computational approaches to regulatory intermediation more effective.

16.2 SOFTWARE-MEDIATED NEGOTIATION

Using software to automate and negotiate consent and privacy preferences is not new. The two most high-profile attempts were Do Not Track (DNT) and the Privacy Preferences Platform (P3P) (both coming out of working groups at the World Wide Web Consortium). DNT is a simple binary value sent in an HTTP header that signals to a website whether or not the user wants to be tracked. P3P was a protocol that allowed websites to express what data they collected in a machine-readable format such that it could be compared to user preferences sent in the HTTP header and applied automatically when the website was served. Neither of these approaches were successful, mainly because both relied on the self-regulation of stakeholders that have little to no incentive to respect the signals sent by users.

To regain control over their personal data, a variety of browser extensions made by web users or privacy organisations have been created. PrivacyBadger – a browser extension developed by the Electronic Frontier Foundation that blocks third party trackers across websites – was developed specifically to address the problem of websites not respecting a DNT signal. Ghostery similarly protects against tracking by blocking HTTP requests of blacklisted origins. While these and similar extensions have the same aim of shifting control from the company collecting data to the data subject, they focus on the technological infrastructure of data processing rather than the user-centric interfaces that have appeared either as a result of the ePrivacy directive or the General Data Protection Regulation.

Browser extensions that do specifically deal with the consent pop-ups predominately focus on improving the users' experience of the website by removing the notifications, rather than improving user control over their personal data. Cookie Notice Blocker (3639 users on Chrome)[2] and Consent Manager (1885 users)[3] both use injected Javascript to block notices, but removing HTML elements does not protect against tracking when you are opted-in by default. We only found one example of a tool that tries to improve the user experience of consent pop-ups but also improve the user's control over the data that is collected on them by interacting with the pop-ups: autoconsent by Sam Macbeth from Cliqz[4]. However, it only provides users with the binary choice of opting in or out, so it does not offer the user granular control over data collection; they might want to allow some things (e.g., language localisation) but reject others (e.g., personalised advertising).

[2] https://chrome.google.com/webstore/detail/cookie-notice-blocker/odhmfmnoejhihkmfebnolljiibpnednn

[3] https://chrome.google.com/webstore/detail/consent-manager/gpkoajillfmlpnglbagpplnphadbfalh

[4] úrlhttps://github.com/cliqz-oss/autoconsent

16.3 INTEROPERABILITY

Interoperability is "[t]he capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units",[5] where the user refers to both human and nonhuman entities. A distinction is made between *syntactic* and *semantic* interoperability, with the former referring to the exchange of data and the latter to the meaningful use of that data.[6] Cory Doctrow, writing for the Electronic Frontier Foundation, has dedicated much of his writing in 2019 to conceptualising and historicising interoperability in the tech industry (see[7] for an overview). In particular, he argues for a renewed interest in *adversarial interoperability*: the efforts of one system to interface with another against the explicit wishes of its progenitors (e.g., Apple's reverse engineering of Microsoft's .doc format so Pages could open them);

Interoperability – whether conceptually, technologically, or ethnographically – does not appear to be a key topic of interest for HCI research. Querying for the term in the entire corpora of CHI, UIST, CSCW, GROUP, ECSCW, TOCHI, and JCSCW returns 288 results. Filtering for only those publications with "interoperability" or "interoperable" in the title, abstract, or author keywords (and removing workshop papers and false positives) reduces that to a mere 30. Almost half mention interoperability tangentially, either as the problem statement (6) or as an implication for design (6). Twelve papers present a system which includes (semantic or syntactic) interoperability to varying degrees (i.e., as its main feature, or as an unintentional side-effect). Four papers discuss interoperability conceptually, all in the context of supporting collaboration and cooperation.

16.4 CONSENT MANAGEMENT PLATFORM DESIGNS

Consent management platforms (CMPs) – what we have been referring to as consent pop-ups – are (third party) services which help websites keep track of the advertising mechanisms they employ on the website, and capture the consent that visitors give about the data that needs to be collected for those advertising mechanisms. Although their overal functionality and visual design are quite similar, their technical implementations are quite different, and affect how they support or inhibit automated interoperability.

16.4.1 *Dynamic vs. static HTML*

Most CMPs load the content of the pop-up dynamically as the user interacts with the interface – only Cookiebot is entirely present in the document object model (DOM) on pageload. Whether all the elements are there on pageload or asyn-

[5] ISO/IEC 2382-01 (1993). *Information Technology Vocabulary, Fundamental Terms*.

[6] Kim H Veltman (2001). 'Syntactic and semantic interoperability: new approaches to knowledge and the semantic web.' In: *New Review of Information Networking* 7.1, pp. 159–183.

[7] Cory Doctorow (2019). 'Adversarial Interoperability.' In: *Electronic Frontier Foundation*. URL: https://www.eff.org/deeplinks/2019/10/adversarial-interoperability.

chronously generated instead makes a big difference for interoperating with it, because it requires a mechanism to continuously detect whether elements exist and handling the time between those checks. This bottleneck puts an upper limit to how fast automated interactions with the interface can be. TrustArc in particular makes a lot of remote calls to retrieve e.g., the vendors that are active on the page and whether they can be opted-out via the CMP or require going to a separate page.

Static HTML – or more precisely, HTML which is static after the page finishes loading – makes it more straight-forward to target elements. Interaction flows in Cookiebot, for example, are managed by simply hiding or displaying elements in the DOM. What makes interoperating with this approach slightly tricky is that the system needs to check whether the elements are actually displayed and visible in order to know whether they are the ones that need to be interacted with. A unique benefit is that often the adversarial system has access to more functionality and could thus outperform even the best user, just by virtue of being able to select options that are never visually represented. Cookiebot, again, has a "Use necessary cookies only" button (their equivalent of declining all) which is almost always hidden but can be clicked programmatically.

16.4.2 *Semantic markup*

A key part of interoperating with another system is being able to accurately and reliably target objects of interest. Semantic information which describes the object's meaning is one key way to facilitate this. The target elements in these CMPs are the buttons, sliders, checkboxes, and toggles that set and submit the consent preferences. In HTML, explicit semantics are added using tags, classes, ids, and attributes.

The quality of the semantic markup varies significantly between different CMPs. Cookiebot has very descriptive ids for the different data processing purpose checkboxes (e.g., `CybotCookiebotDialogBodyLevelButtonMarketing`), while QuantCast, Crownpeak, OneTrust, and TrustArc use non-specific class names such as `category-check` for all of them. Certain versions of OneTrust even have factually incorrect semantics: purposes and vendors are placed under the `menu-item-necessary` category (which they do not need consent for) yet are in fact optional. Some CMPs use auto-generated classes, making it impossible to consistently interoperate with it.

If no semantic markup is available, it can be inferred through the hierarchical structuring in the DOM tree of elements that logically belong together (e.g text labels and toggles). The DOM tree is volatile, however, and inferring meaning by traversing its structure is not very robust. For example, QuantCast separates purposes and vendors into different, dynamically generated pages, making it impossible to figure out how they relate to each other.

The only recourse left is analysing nearby text content to understand what the element contains, but this creates language processing problems such as typos and localisation differences.

16.4.3 *Hidden state*

Proper software engineering principles argues for encapsulation of data within methods that use it, and against storing state in something such as the DOM. Indeed, most of the CMPs store the state entirely as JavaScript runtime state. However, this makes it difficult for an (adversarially) interoperating system to verify that the changes it effects are recorded, stored, and submitted. For example, although toggling a checkbox might add or remove a class, it does not follow that adding or removing that class changes the recorded value of a checkbox. The CMP might instead be using an `eventListener` on a click to edit the consent value. As a result, interoperating with such a system precludes making direct changes to the data and instead requires simulating user interactions.

16.5 CONSENT-AUTOMATING SOFTWARE: CONSENT-O-MATIC

Our goal is to automate the interaction between a data subject and a consent pop-up as a way to deal with their (at best) confusing or (at worst) intentionally manipulative interface designs. The design of the extension and the mechanisms that needed to be created were shaped by the way the CMPs are implemented as described above. The extension is published on the Chrome[8] and Firefox[9] web-store (but built using the WebExtensions API so should also work for Opera and Edge) and the code is available on Github[10].

Rules for how to interoperate with a CMP are stored in JSON files in a declarative notation (fig. 53), which makes it easy to add new CMPs to the extension or edit existing instructions. The basic structure of the JSON consists of two main parts: `detectors` and `methods`. `detectors` are used to determine a CMP is present in the DOM and visually showing on the page by checking for the existence of a particular HTML element that is specified by the user. If the `detector` returns a positive result, the `methods` are executed, which can be one of three (optional) actions: `OPEN_OPTIONS`, `DO_CONSENT`, or `SAVE_CONSENT`.

16.5.1 *DOM Selection and Actions*

The building blocks of these two parts of the system are a *DOM Selection* mechanism that is more sophisticated than what is normally possible with CSS selectors, and a list of *Actions* that can be used to trigger events on particular elements. The *DOM Selection* works by first using a CSS `selector` to target elements and subsequently applying (optional) filters to what is returned to further winnow down the list (see 12 for the five types of filters).

Because the way some CMPs are implemented make it very difficult to target certain elements, it is possible to execute this selection process in two steps using the `parent` and `target` construction. The selection and filtering options are the

[8] https://chrome.google.com/webstore/detail/consent-o-matic/mdjildafknihdffpkfmmpnpoiajfjnjd
[9] https://addons.mozilla.org/en-US/firefox/addon/consent-o-matic/
[10] https://github.com/cavi-au/Consent-O-Matic

```
{"myCMP": {
    "detectors": [
        {   "presentMatcher": {
                "type": "css",
                "target": {
                    "selector": "#theCMP"}}}],
    "methods": [
        {   "name": "OPEN_OPTIONS",
            "action": {
                "type": "click",
                "target": {
                    "selector": ".button",
                    "textFilter": "Change settings"}}},
        {   "name": "DO_CONSENT",
            "action": {
                "type": "list",
                "actions": [
                    {   "type": "click",
                        "target": {
                            "selector": ".menu-vendors"}},
                    {   "type": "consent",
                        "consents": [
                            {   "type": "A",
                                "matcher": {
                                    "type": "checkbox",
                                    "parent": {
                                        "selector": ".vendor-item",
                                        "textFilter": "Functional cookies"},
                                    "target": {
                                        "selector": "input"}},
                                "toggleAction": {
                                    "type": "click",
                                    "parent": {
                                        "selector": ".vendor-item",
                                        "textFilter": "Functional cookies"},
                                    "target": {
                                        "selector": "label"}}}]}]},
        {   "name": "SAVE_CONSENT",
            "action": {
                "type": "click",
                "target": {
                    "selector": ".save-consent-btn"}}}]}}
```

Figure 53. Dummy example of a JSON ruleset for a particular CMP (brackets condensed to preserve space)

Filter type	Description
`textFilter`	filters all nodes that do not include the given (array of) string(s).
`styleFilter`	filters based on `computedStyle`. Takes an `option` which is the CSS property name, and a `value`. Can be `negated` `True` or `False` to specify presence or absence.
`displayFilter`	filters based on if the element is displayed or not.
`iframeFilter`	filters based on if a node is inside an iframe or not (Boolean).
`childFilter`	filters based on whether the node has a particular child element, which is specified using the same DOM Selection process.

Table 12. The five types of filters that can be used to select the element of interest.

Action type	Description
`click`	simulates a click.
`slide`	simulates moving a slider.
`hide`	adds `display: none`.
`close`	closes the current tab (for when new ones are opened).
`ifcss`	conditionally executes an action based on whether an element is or is not found.
`waitcss`	waits until it can or cannot find the target element after a specified `waitTime` for the number of `retries`.
`foreach`	runs the same action for each of the elements found.
`consent`	takes a list of the user's consent choices (see below) and tries to toggle an element value based on that.
`list`	takes a list of actions to apply to the same element.

Table 13. The nine types of actions that can be executed on an element of interest.

same for both, but this allows users to first select the "root" parent node and subsequently select one or multiple targets relative to that node.

Actions are what the Consent-o-Matic extension executes once a target has been selected. Some actions are for manipulating a target element, other actions are about controlling the flow of interactions with the CMP (see table 13 for the nine types of actions).

16.5.2 *Consent Preferences*

The user can specify their privacy preferences by toggling six different data processing purposes in the extension's settings page (fig. 54). These purposes are based on a survey of the purposes found in 680 CMPs used by the top 10.000 most popular websites in the UK.[11] Different spellings of the purpose category names were first collapsed by using OpenRefine[12]. Then, the descriptions of those different purposes were compared and further merged until six separate ones remained: Information Storage and Access; Preferences and Functionality; Performance and Analytics; Content selection, delivery, and reporting: Ad selection, delivery, and reporting; and Other Purposes. The purposes found in the CMPs we interoperate with are matched to these global data processing categories and the user's answer for each of them (opt-in or opt-out) are passed to the consent action mentioned previously.

16.6 NEGOTIATION OUTCOME

As of July, nearly 4000 people have installed the *consent-automating extension* on Chrome and Firefox (fig. 55). Since the launch on December 24th, five external contributors have joined the project, and collectively expanded the ruleset from five to now cover thirty-five different pop-up designs. Sam Macbeth, developer

[11] Midas Nouwens et al. (2020c). 'Dark Patterns Post-GDPR: Scraping Consent Interface Designs and Demonstrating their Influence.' In: *Conditionally accepted for CHI 2020*. ACM.

[12] An application used for data wrangling that has various fuzzy matching algorithms.

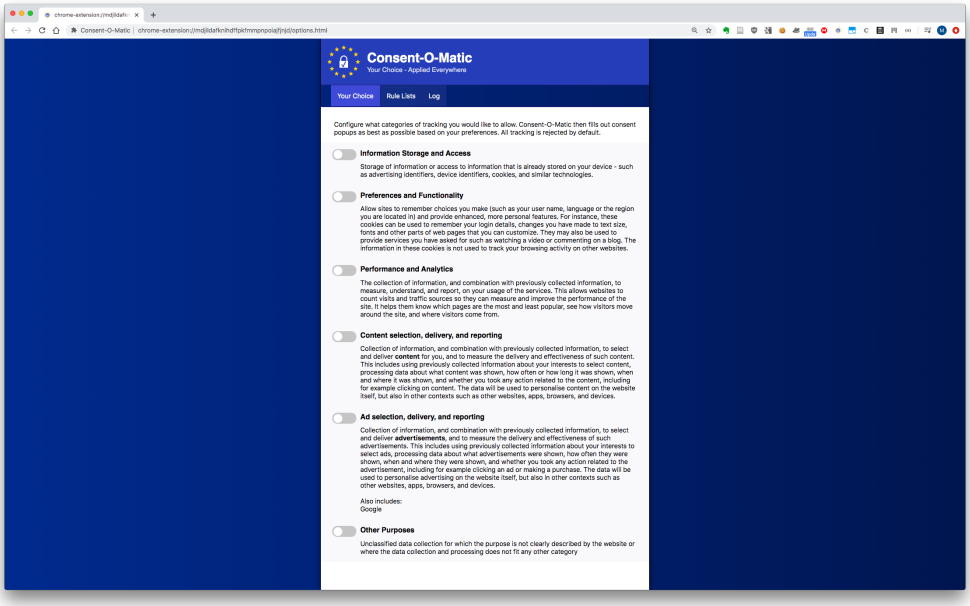


Figure 54. Consent-O-Matic's data processing purposes that can be toggled.

for the privacy-preserving Cliqz browser, added support for the Consent-O-Matic extension to their own autoconsent plugin, allowing the JSON rules to be used in both (unfortunately, Cliqz has since then ceased operations). The extension has been mentioned by a handful of the news articles that reported on the study described in Chapter 14, as well as received word-of-mouth recommendations on forums such as HackerNews and Reddit.

How successfully does this negotiation software allow users to change the interaction design of consent pop-ups? Without a controlled experiment comparing the manual preference settings with the automated consent, it is difficult to say whether the use of the extension results in people feeling more empowered. Based on online discussions and anecdotal feedback, however, it has made people feel more in control of their data and help them get closer to their preferred web browsing experience.

Apart from the experiential impact, there is also some ambiguity around how well it lets users enforce their data protection preferences.

There is no guarantee that the consent answer submitted via the extension is actually honoured by the website. Because browsers sandbox extensions for security reasons, they do not have access to the internal JavaScript runtime state of the page, nor the (encrypted) cookies set in the client header. As a result, it is not possible to verify what consent settings are ultimately applied. This means that the extension only works as well as a really informed (and fast) user, but cannot guard against malintent from the website provider [13].

The extension is also not incredibly robust, because it interoperates with pop-ups adversarially. The extension uses the semantics or structure of DOM elements

[13] Upsettingly, Matte et al.(Matte, Bielova, and Santos, 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework') show that roughly 7% of websites do not respect the users' pop-up answers.

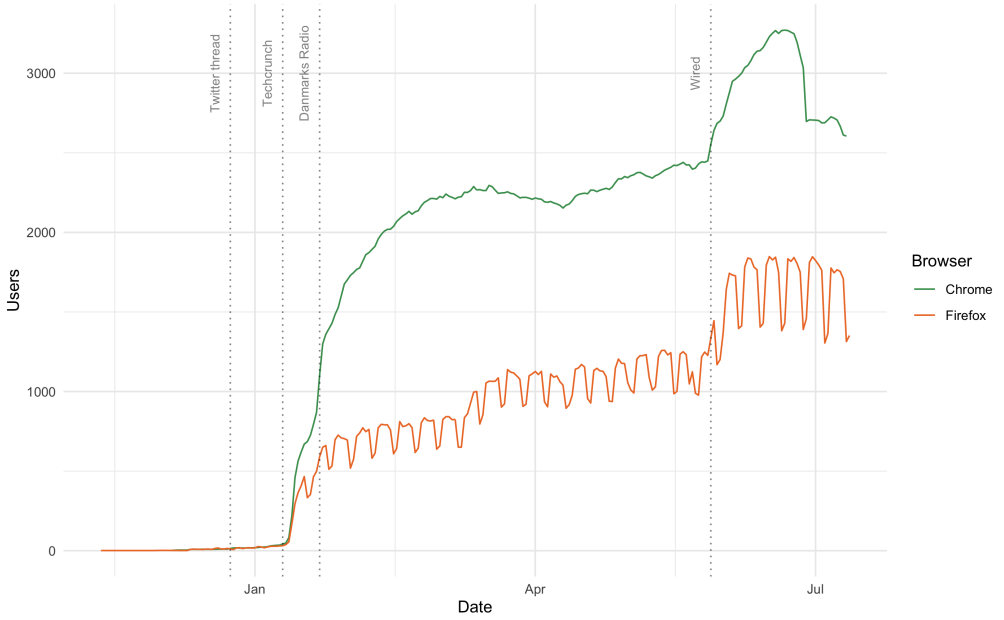


Figure 55. Number of users of the extension per browser. Chrome counts enabled installs, Firefox counts active use (thus dips during the weekend).

to work, so it will break as soon as the pop-up provider changes its technical implementation. However, the way these third-party pop-ups are deployed, and the network in which they exist, create a certain stability. First, the Interactive Advertising Bureau – the advertising industry's most prominent association – has created their own standardised set of processing purposes, so websites can (illegally) pass around consent for certain purposes between websites through a "global consent" signal. If the advertising industry decided to change their purposes, it would force us to update the hard-coded connection between the extension's purposes that users can toggle and their equivalent in the CMP. But it would also require the pop-up companies to do the same if they wanted to keep using the global consent mechanism and connect a user's consent on one site to a similar purpose on another. Second, the scripts which inject the pop-up are deployed in a decentralised way: most websites host the Javascript file on their own server or directly copy it into their HTML. This means that a coordinated, centralised change to the code is impossible, making it harder for pop-up companies to unilaterally break our interoperating code. Even if they would centralise the Javascript, many websites have customised the code to better fit with their branding. Making changes to interfere with our code would also inconvenience their revenue-generating clients, something they are unlikely to do.

16.7 CONCLUSION

The interaction design of consent pop-ups can be changed by adversarially interoperating with it using consent-automating negotiation software.

Although this chapter does not provide empirical measurements of the impact of the Consent-O-Matic extension, it is not unreasonable to suggest that changing

the way users provide consent from a website-level to a browser-level activity improves their experience of the web, and results in a more consistent application of their data protection preferences. Rather than being faced with a barrage of notices that all have their own manipulative designs, people can set their preferences once and rely on public servants with the knowledge and resources to develop software that can interoperate with pop-ups – my colleagues and I at Aarhus University – to represent their interests. Of course, this approach is a fragile fix for a poor interpretation of data protection law. If the consent signals submitted through browser settings better reflect user's *real* data protection preferences than those submitted using pop-ups on websites, then a browser extension that people need to voluntarily install is not the best way for us to switch between these models. Rather, data protection choices should be part of the default settings of the browser, and the consent signal received by a website through them should be legally binding.

While browser settings are currently not considered a legally valid way to collect consent under the GDPR, this mechanism *was* proposed in the ePrivacy Regulation, set to replace the 2002 ePrivacy Directive. The original draft of the regulation submitted by the European Commission in 2017 included the requirement that "[s]oftware placed on the market permitting electronic communications, *including the retrieval and presentation of information on the internet*, shall offer the option to prevent third parties from storing information on the terminal equipment of an end-user or processing information already stored on that equipment" (emphasis added).[14] The idea was that when someone installed a browser, "the software shall inform the end-user about the privacy settings options and, to continue with the installation, require the end-user to consent to a setting".[15] The Article 29 Working Party – the official advisory body consisting of a representative of each Member State's DPA – supported these requirements and "strongly recommend[ed] to make adherence to the Do Not Track standard mandatory".[16]

Unfortunately, the article was lobbied out during the latest round of amendments. The ePrivacy Regulation has been languishing for years, passed around between presidencies of the Council of the EU like a hot potato. The six-month German presidency (1 July–31 December 2020) reopened negotiations recently, adding a COVID-19 flavour by asking whether the "vital interest" of a data subject (i.e., their health) is a legally valid reason for reading or writing information from their devices.[17] Regrettably, the question whether preferences expressed via browser settings should be legally binding has not made a reappearance. As such, it seems we are stuck with the website-level notice and consent model for a while longer, and developing negotiation software that lets an individual take direct action against manipulative interfaces will be imperative until a more robust regulatory intervention materialises.

[14] European Commission, *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Communications)*.

[15] Ibid.

[16] Article 29 Working Party (2017). *Opinion 01/2017 on the Proposed Regulation for the ePrivacy Regulation (2002/58/EC)*. European Commission.

[17] Council of the European Union (2020). *Interinstitutional File: 2017/0003(COD)*.

17.

CONCLUSION

It took (Denmark) two centuries to make industrial capitalism (mostly) safe for democracy and a dignified life by establishing fundamental rights and protections (e.g. minimum wage, healthy and safe working conditions, collective agreement-based governance). Over the past thirty years, global regimes have been struggling to do the same with informational capitalism. With its orientation towards information as the main source of surplus value, our institutions, labour, culture, and economies have transformed themselves to better accumulate more data, but also make sure humans produce more data. When asked how much progress the European Union had made to ensure we did not "walk naked" into this new form of capitalism, European Commissioner Margrethe Vestager replied: "I'm afraid we only have a thong on".[1]

The multi-stakeholder struggle over the web described in part II of this dissertation is a microcosm of the ideological battle for the digital future. In Chapter 13, I chronicled how the web's original technolibertarian design was quickly subverted by the data-centric interests of commercial companies, and tracking technologies such as cookies were baked into its fundamental protocols. The push-back by civil organisations, governments, courts, and international bodies established the "notice and consent" governance mechanism to protect human autonomy, now further entrenched through the EU's General Data Protection Regulation. Regrettably, as I have shown in Chapter 14, this mechanism continues to fall short, as more than 80% of these consent pop-up do not comply with the interface guidelines required for the consent to be legally valid: they use dark patterns such as pre-checked permission boxes, count implicit behaviour such as scrolling as explicit consent, and make it significantly harder to reject than to accept tracking. Data protection authorities tasked with regulating this ecosystem have struggled to push back, so it is encouraging to see that direct action based on data gathered by the automated compliance monitoring tool described in Chapter 15 can be extremely effective. Rgulatory action in response to the lack of compliance highlighted by this study has helped increase the rate of bulk consent buttons from 1,5% to 41% between February and July 2020. Similarly promising, Chapter 16 describes how, even if authorities fail to regulate, nearly four thousand

[1] Ingeniørforeningen (IDA) (2020). *Shoshana Zuboff meets Margrethe Vestager: A conversation about a future digital Europe - webinar*. URL: https://ida.dk/arrangementer-og-kurser/arrangementer/shoshana-zuboff-meets-margrethe-vestager-a-conversation-about-a-future-digital-europe-webinar-336843 (visited on 07/10/2020).

individuals can successfully subvert dark patterns and enforce their preferences through the use of consent-automating browser extensions.

I have successfully used software to negotiate the design of web tracking technology in the context of European data protection regulation. Yet throughout this process it has been impossible to ignore the fragility of the EU's governance approach, and the seeming improbability that it will actually produce systemic, lasting change. Data protection has been touted as a preeminent instrument to control the power of technology multinationals and protect European digital sovereignty. Steeped in the human rights movement that emerged after World War II, the explicit objective of the European Commission for data protection regulation has been to "strengthen individual's rights" and "enhance control over one's own data". This means that the EU has premised a considerable part of the GDPR's success on the idea of the rational, informed, and autonomous liberal subject.[2] It elects to believe that global tech behemoths with almost infinite resources can be held to account by people exercising their rights; that the thundering course of informational capitalism can be corrected by the uncoordinated and underpowered actions of individuals. The notice and consent model is the perfect representation of this ideology: it has placed the responsibility of controlling the incredibly complex ways in which data is collected and processed with the individual, now forced to continuously make "informed" and "specific" decisions every time they visit a website.

Unsurprisingly, ignoring a power asymmetry does not make it go away, and people have not felt more empowered by the rights conferred to them via data protection law. Between 2015[3] and 2019,[4] people's perceived control over their data has not meaningfully changed at all. If anything, after the GDPR, *fewer* people feel in complete control (from 15% to 14%). The individualistic approach to data protection might not only be ineffective, but actually directly undermine the collective benefits it tries to create. When individuals use consent mechanisms to reject tracking, it makes it easier for companies to identify those who did not.[5] In other words, the benefits gained by some come at the cost of the privacy of others. This operationalisation that emphasises individual responsibility appears unlikely to achieve governance at scale.

The *negotiation software* developed in this dissertation has tried to address the asymmetric power distribution over tracking technologies and advance a more

[2] Linnet Taylor (2020). 'Public actors without public values: legitimacy, domination and the regulation of the technology sector.' In:

[3] European Commission and Directorate-General for Justice and Consumers and TNS Opinion Social (2015). *Data protection: report*. European Commission. ISBN: 978-92-79-48414-8. URL: http://dx.publications.europa.eu/10.2838/552336.

[4] European Commission and Directorate-General for Justice and Consumers and European Commission and Directorate-General Communication and Kantar (2019). *The General Data Protection Regulation: report*. ISBN: 978-92-76-08384-9. URL: https://data.europa.eu/doi/10.2838/579882.

[5] Guy Aridor, Yeon-Koo Che, and Tobias Salz (2020). *The Economic Consequences of Data Privacy Regulation: Empirical Evidence from GDPR*. ID 3522845. DOI: 10.2139/ssrn.3522845. URL: https://papers.ssrn.com/abstract=3522845.

collective approach to data protection. The *automated compliance monitoring* tool changes how consent pop-ups are currently regulated, which requires individual citizens to lodge complaints about specific domains with the data protection authorities to affect a change. Instead, it allows us to monitor industry-wide practices, faster and cheaper than the small-scale methods predominately used by resource-constrained DPAs. The *consent-automating browser extension* takes away the need for people to make complicated, website-level privacy decisions. Instead, it protects against the manipulative design practices of consent management platforms by outsourcing the decision-making process to public servants (my colleagues and I at Aarhus University) who have the domain-knowledge and resources to represent and communicate people's preferences to data controllers and processors.

Negotiation software that collectivises the exercise of individual rights does not fix the way European data protection is operationalised, but it does, in its own way, contribute to the regulation of global tech companies and redistribution of power and control over software.

Last and Final Offer

18.

NEGOTIATING SOFTWARE AS A COUNTERMOVEMENT

Since the 1970s, digital technologies increasingly determine "who gets what, when, and how",[1] and control over those technologies has concentrated in the hands of just a few private corporations.

However, there is cause for hope. Writing about the US American economy in the 1950s, Galbraith argued that the monopolistic power of the large firms would inevitably lead to the manifestation of "countervailing powers" to correct for the imbalances, exploitations, and distortions created by corporate dominance (e.g., trade unions, civil society organisations). Competition as a model for market regulation had failed, he claimed, but the "tendency of power to be organized in response to a given position of power"[2] would ensure that the conditions for a dignified life would eventually prevail. As advisor to John F. Kennedy, he advanced the position that the role of a government – perhaps even the most important one – was to provide the minimum opportunity for a countervailing power to organise itself.

Galbraith was not alone in his belief of corrective dyadic relationships as the engine behind social change. Roughly a decade before, Polayni conceptualised the "double movement" to explain the origins of our current market societies (i.e., capitalism). The first movement was reformers working to expand the role of markets and curb the regulating influence of other social institutions, notably by turning previously uncommodified things such as land, labour, and money into tradeable goods. The destructive implications of this helped foment the second movement – the countermovement – where nation states reinvented themselves as market interventionists and providers of social protections, culminating in the birth of the social democracy.

Ultimately, however, both Polanyian countermovements or Galbraithian countervailance create only temporary equilibria, because they themselves become targets of evasion, co-optation, and redefinition. Julie Cohen, in her book *Between Truth and Power*, describes how informational capitalism has systematically dismantled or worked around the social protections instituted after WWII that immunised workers, consumers, and citizens against the harms of laissez-faire economics. And so, the unchecked power of multinational technology corporations

[1] Lasswell, *Politics: Who Gets What, When, How*.

[2] John Kenneth Galbraith (1952). 'American Capitalism: The Concept of Countervailing Power.' In: *Boston: Houghton Mifflin*, p. 113.

that we are witnessing today requires the invention of a new countermovement, one that will "engage directly with the logics of dematerialization, datafication, and platformization".[3] Cohen cautions, however, that this new countermovement cannot rely on the instruments from the previous economic era, but needs to develop new methods that respond directly to the nature of the information age.

This dissertation has explored two ways to negotiate software as possible countermovements to informational capitalism.

The *negotiable software* described in Part I focused on principle-based technological redesign, making distributed control an inherent quality of our software applications. The computational medium Codestrates demonstrates the technological feasibility of this approach. It unequivocally shows it is possible to make contemporary workplace applications whose code can be changed at a fundamental level from within itself, during run-time, and in a collective and distributed fashion. Such exemplars can function as proof of concepts or rebuttals:[4] they make concrete the possibilities (and limitations) of technology and can showcase a possible alternative future.[5] This helps shift the window of discourse when we critique today's autocratically designed systems, a discourse which might otherwise focus on how achievable or practical negotiability would be as an embedded quality.

However, it also showed the limitations of such a demonstrative approach to negotiating software at scale. Some technologies, such as the internet and the PC, are argued to have generative features that give it the "capacity to produce unprompted change driven by large, varied, and uncoordinated audiences".[6] The dominant model for end-user software, however, has features that arguably to do the opposite: the copyright and intellectual property battles around software in the 90s has created a climate where source code is hermetically sealed away and obfuscated, and the shrink-wrapped designs of software won out against component-based systems with after-market ecosystems of plugins, addons, and macros. Rarely do mass-market software applications provide the infrastructure for "uncoordinated audiences" to augment the software's designs, thus making it difficult for researchers (and any other stakeholder) to deploy code that renegotiates the software people are already using. Instead, they would have to enter the marketplace as a competitor, for which the barriers to entry are substantial (e.g., network externalities, economies of scope, vertical integration[7]), and which is at odds with the incentive structures and funding possibilities of academia.

The *negotiation software* described in Part II focused on process mediation, using digital tools to support existing ways that data-processing technologies are regulated. These software can be used as a direct mediator between two

[3] Cohen, *Between Truth and Power: The Legal Constructions of Informational Capitalism*, p.270.

[4] Abebe et al., 'Roles for computing in social change.'

[5] Salovaara, Oulasvirta, and Jacucci, 'Evaluation of Prototypes and the Problem of Possible Futures.'

[6] Jonathan Zittrain (2009). 'The Generative Internet.' In: *Communications of the ACM* 52.1, 18–20. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1435417.1435426.

[7] Jacques Crémer et al. (2019). *Competition policy for the digital era.* ISBN: 978-92-76-01946-6. URL: http : / / publications . europa . eu / publication / manifestation _ identifier / PUB _ KD0419345ENN.

actors or instead provide more indirect support. For example, the Consent-O-Matic browser extension sits between web users and the consent management platforms (which themselves represent the website owners) and functions as a representative of the users' preferences by directly interacting with the pop-up interface. The web-scraper, on the other hand, represents auxiliary negotiation software that is used as a diagnostic,[8] and whose purpose is to provide data that can be used as leverage when communicating with news media and data protection authorities through other channels. Both types of negotiation software supported actions that made a substantial and seemingly sustainable change to the way people experience consent pop-ups on the Danish web.

However, it also became apparent that this approach is limited in scope: it can support negotiation processes that already exist, but is less suited for creating or imagining new ways of effecting change. The browser extension works because the web is based on open standards and the source code of the target software is accessible and can be interoperated with. The web-scraper works because there are rights-based mechanisms that can be exploited and regulatory bodies that have a duty to respond. Building digital tools that facilitate a process for which there is no precedence, no legal duty, or which has little support from other stakeholders is unlikely to be as effective.

The strength of *negotiable software* is that it allows us to demonstrate technical feasibility, but it is limited in the scale at which these alternatives can be deployed and directly affect the status quo. The strength of *negotiation software* is that it can help us maximise the impact of existing processes of change, but is limited in its scope because it is best used to support paths to change that already exist in society. As such, these principle-based and process-based countermovements should be seen as complementary, rather than substitutions, and future work should focus on a two-pronged approach.

For example, in the domain of workplace applications, negotiation software could be used to diagnose digital labour conditions and bring to attention the way that existing regulation is not being complied with. European health and safety directives from the 1990s, transposed into national law by all member states at least since 1992, already demand that software must be "suitable for the task", "adaptable to the operator's level of knowledge", and that no "quantitative or qualitative checking facility may be used without the knowledge of the workers".[9] All Danish employers are legally required to evaluate their working conditions at least once every three year, yet my preliminary research shows that these requirements around the quality of the software – inscribed in their labour law – are seemingly never included. As a result, any software-related harms Danish workers experience in one of the most digitalised countries in the world are utterly unrecorded. When notifying local labour representatives, the response was that addressing non-compliance "would require a lot of work and the likelihood of succeeding seems slim"; the response from the IT-political consultant of

[8] Abebe et al., 'Roles for computing in social change.'

[9] Council of European Union (1990). *Directive on the minimum safety and health requirements for work with display screen equipment (90/270/EEC)*. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31990L0270.

Denmark's IT union was that they "never had any member raise regulation of the software they use as an issue – not even informally, so I believe that is not a big issue". In both cases, negotiation software that could demonstrate harm and non-compliance might be used to convince these representatives and activate their considerable leverage to negotiate changes with employer delegates.

Similarly, in the context of data protection and digital rights, negotiable alternatives of existing software could be designed that showcase what an ecosystem amenable to distributed control would look like. For example, consent pop-ups could be redesigned to include interoperability mechanism such as public APIs that allow third-party code to configure and monitor their implementations. Data protection authorities might benefit from web-scraper technologies packaged inside a negotiable system such as Codestrates, allowing them to adjust and tweak its design for a variety of automated regulatory purposes.

While different in some respects, both principle-based negotiable software and process-based negotiation software revealed the limitations of individualistic strategies. Successful changes to a software's design described in this dissertation happened primarily in situations where collective power was used – for example when user cooperatives stood up against computer manufacturers; when nanoscientists leveraged colleagues, friends, and even us researchers to debug and adapt their scripts; when web-users relied on our Consent-O-Matic browser extension to change their interaction with consent pop-ups; or when I instrumented news media to apply pressure to data protection authority to in turn use their power to regulate non-compliant data controllers. Ultimately, the capacity of the liberal subject to stand up against corporate power is limited, and more *collective* countermovements need to be developed.

Denmark is uniquely positioned to explore this angle to negotiating software. It has one of the highest levels of trust in national institutions in Europe,[10] unions are still some of the most influential organisations in the country,[11] and adherence to the rule of law is the strongest in the world.[12] Denmark was also the first country to create the Office of Technology Diplomacy, and established embassies in Silicon Valley and Beijing to make sure the country's interests were represented in connection with the software developed there (although the ambassador has recently quit and started working for Microsoft). It was the first country to see a collective agreement between a union and a platform app (although the app's design was not part of that negotiation). Rather than relying on individuals to reprogram their software or chase after foreign tech giants, these responses to the digitalisation of their society demonstrate their commitment to using collective power for achieving universal welfare. The Nordic collectivist model has managed to make industrial capitalism safe for and compatible with an egalitarian and solidary society. Perhaps it can do the same for informational capitalism.

[10] Eurofound (2018). *Societal change and trust in institutions*. URL: https://www.mm.dk/misc/Democracy-Perception-Index-2018-1.pdf, p. 16.

[11] Markus Bernsen, Christoph Ellersgaard, and Anton Grau Larsen (2015). *Magteliten: Hvordan 423 danskere styrer landet*. Politikens forlag.

[12] World Justice Project (2020). *Rule of Law Index*. URL: https://worldjusticeproject.org/sites/default/files/documents/WJP-ROLI-2020-Online_0.pdf, p. 16.

BIBLIOGRAPHY

'95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data' (1995). In: *Official Journal of the EC* 23.6.

Aagaard, Emilie (2020). 'Forskere: DR's og Folketingets hjemmesider er på kant med persondataloven.' In: *Danmarks Radio*.

Aarhus University Graduate School of Arts (2012). *Rules for the PhD Programme at the Graduate School, Art*. https://phd.arts.au.dk/fileadmin/phd.arts.au.dk/AR/Generelle_retningslinjer_UK_1-11-2012.pdf.

Abebe, Rediet, Solon Barocas, Jon Kleinberg, Karen Levy, Manish Raghavan, and David G Robinson (2020). 'Roles for computing in social change.' In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 252–260.

Acquisti, Alessandro and Jens Grossklags (2005). 'Privacy and rationality in individual decision making.' In: *Security Privacy, IEEE* 3.1, pp. 26–33.

Acquisti, Alessandro et al. (Aug. 2017). 'Nudges for Privacy and Security: Understanding and Assisting Users' Choices Online.' In: *ACM Comput. Surv.* 50.3, 44:1–44:41. ISSN: 0360-0300. DOI: 10.1145/3054926. URL: http://doi.acm.org/10.1145/3054926.

Adar, Eytan, David Karger, and Lynn Andrea Stein (1999). 'Haystack: Per-user Information Environments.' In: *Proceedings of the Eighth International Conference on Information and Knowledge Management*. CIKM '99. Kansas City, Missouri, USA: ACM, pp. 413–422. ISBN: 1-58113-146-1. DOI: 10.1145/319950.323231. URL: http://doi.acm.org/10.1145/319950.323231.

Advocate General Szupunar (2019). *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:246, Opinion of the Advocate General*.

Adzerk (2019). *Adtech Insights — August 2019 Report*. URL: https://adzerk.com/assets/reports/AdTechInsights_Aug2019.pdf.

Akera, Atsushi (2001). 'Voluntarism and the Fruits of Collaboration: The IBM User Group, Share.' In: *Technology and Culture* 42.4, 710–736. ISSN: 0040-165X.

Alkhatib, Ali, Michael S. Bernstein, and Margaret Levi (2017). 'Examining Crowd Work and Gig Work Through The Historical Lens of Piecework.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, pp. 4599–4616. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025974. URL: http://doi.acm.org/10.1145/3025453.3025974.

Angulo, Julio, Simone Fischer-Hübner, Tobias Pulls, and Erik Wästlund (2011). 'Towards Usable Privacy Policy Display & Management for PrimeLife.' In: *S. M. Furnell, & N. L. Clarke (Eds.), Proceedings of international symposium on human aspects of information security & assurance (HAISA 2011)*, pp. 108 –117.

Apple Computer (1983). 'It took 200 years to develop programs you can learn in 20 minutes.' In: *Personal Computing* 6.

Apple Computer (1995). 'Macintosh vs. Windows 95: OpenDoc.' In: URL: http://tech-insider.org/mac/research/acrobat/Mac/950829.pdf.

Arbejdstilsynet (n.d.). *The working environment legislation*. URL: https://at.dk/en/regulations/working-environment-legislation/ (visited on 02/22/2021).

Aridor, Guy, Yeon-Koo Che, and Tobias Salz (2020). *The Economic Consequences of Data Privacy Regulation: Empirical Evidence from GDPR*. ID 3522845. DOI: 10.2139/ssrn.3522845. URL: https://papers.ssrn.com/abstract=3522845.

Armer, Paul (1956). 'SHARE - A Eulogy to Cooperative Effort.' In: *Annals of the History of Computing 2*.

Article 29 Working Party (2018). *Guidelines on Consent under Regulation 2016/679 (WP259 rev.01)*. European Union.

Atlassian Confluence (2019). URL: https://atlassian.com/confluence.

Autoriteit Persoonsgegevens (2019). *Hoe Legt de AP de Juridische Normen Rond Cookiewalls Uit?* Den Haag: AP.

Ayyagari, Ramakrishna, Varun Grover, and Russell Purvis (2011). 'Technostress: Technological antecedents and implications.' In: *MIS quarterly*, pp. 831–858.

B., Meinert David, Dane K. Peterson, John R. Criswell, and Martin D. Crossland (2006). 'Towards Usable Privacy Policy Display & Management for PrimeLife.' In: *Journal of Electronic Commerce in Organizations (JECO)* 4.1, pp. 1–17.

Badam, Sriram Karthik, Andreas Mathisen, Roman Rädle, Clemens N. Klokmose, and Niklas Elmqvist (2018). 'Vistrates: A Component Model for Ubiquitous Analytics.' In: *IEEE Transactions on Visualization and Computer Graphics*. ISSN: 10772626. DOI: 10.1109/TVCG.2018.2865144. URL: https://karthikbadam.github.io/assets/data/vistrates.pdf.

Bardram, Jakob, Jonathan Bunde-Pedersen, and Mads Soegaard (2006). 'Support for Activity-based Computing in a Personal Computing Operating System.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: ACM, pp. 211–220. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124805. URL: http://doi.acm.org/10.1145/1124772.1124805.

Barney, Doug (1993). 'Object Linking Readied for Unix Notes Clients, Programs.' In: *InfoWorld* 15.24.

Berg-Beckhoff, Gabriele, Grace Nielsen, and Eva Ladekjær Larsen (2017). 'Use of information communication technology and stress, burnout, and mental health in older, middle-aged, and younger workers–results from a systematic review.' In: *International journal of occupational and environmental health* 23.2, pp. 160–171.

Bernsen, Markus, Christoph Ellersgaard, and Anton Grau Larsen (2015). *Magteliten: Hvordan 423 danskere styrer landet*. Politikens forlag.

Bidgoli, Hossein (2004). 'The internet encyclopedia (Volume 2).' In:

Bier, Eric A. (1991). 'EmbeddedButtons: Documents As User Interfaces.' In: *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*. UIST '91. Hilton Head, South Carolina, USA: ACM, pp. 45–53. ISBN: 0-89791-451-1. DOI: 10.1145/120782.120787. URL: http://doi.acm.org/10.1145/120782.120787.

BIOVIA (2019). URL: https://3dsbiovia.com.

Bødker, Susanne (2006). 'When Second Wave HCI Meets Third Wave Challenges.' In: *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles*. NordiCHI '06. Oslo, Norway: ACM, pp. 1–8. ISBN: 1-59593-325-5. DOI: 10.1145/1182475.1182476. URL: http://doi.acm.org/10.1145/1182475.1182476.

Bødker, Susanne and Morten Kyng (2018). 'Participatory design that matters—Facing the big issues.' In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 25.1, pp. 1–31.

Borowski, Marcel, Roman Rädle, and Clemens N. Klokmose (2018). 'Codestrate Packages: An Alternative to "One-Size-Fits-All" Software.' In: *CHI EA '18 Proceedings of the 2018 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. DOI: 10.1145/3170427.3188563.

Bösch, Christoph, Benjamin Erb, Frank Kargl, Henning Kopp, and Stefan Pfattheicher (2016). 'Tales from the dark side: Privacy dark strategies and privacy dark patterns.' In: *Proceedings on Privacy Enhancing Technologies* 2016.4, pp. 237–254.

Bossen, Claus, Christian Dindler, and Ole Sejer Iversen (2012). 'Impediments to user gains: experiences from a critical participatory design project.' In: *Proceedings of the 12th Participatory Design Conference: Research Papers-Volume 1*, pp. 31–40.

Boyatzis, Richard E. (1998). *Transforming Qualitative Information: Thematic Analysis and Code Development*. SAGE Publications, Inc. ISBN: 0761909613.

Brandel, Mary (1999). '1955: IBM customers form the first computer user group.' In: *CNN*. URL: http://edition.cnn.com/TECH/computing/9905/05/1955.idg/.

Braun, Virginia and Victoria Clarke (2006). 'Using thematic analysis in psychology.' In: *Qualitative Research in Psychology* 3.2, pp. 77–101. DOI: 10.1191/1478088706qp063oa. eprint: http://www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa. URL: http://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa.

Brave (2020). *Europe's governments are failing the GDPR*.

Brinkley, Ian, Michelle Mahdon Rebecca Fauth, and Sotiria Theodoropoulou (2009). *Knowledge workers and knowledge work: A knowledge economy programme report*. Work Foundation.

Brown, Brown and Kenton O'Hara (2003). 'Place as a Practical Concern of Mobile Workers.' In: *Environment and Planning A* 35.9, pp. 1565–1587. DOI: 10.1068/a34231.

Brown, Ian and Christopher T Marsden (2013). *Regulating code: Good governance and better regulation in the information age*. MIT Press.

Brown, Monique R. (1998). 'Revenge of the cookie monster.' In: *Black Enterprise*, pp. 42–44.

Bruns, Axel (2019). 'After the 'APIcalypse': Social Media Platforms and Their Fight against Critical Scholarly Research.' In: *Information, Communication &*

Society 22.11, pp. 1544–1566. DOI: 10.1080/1369118X.2019.1637447. URL: https://doi.org/10.1080/1369118X.2019.1637447.

Bucher, Tania (2013). 'Objects of Intense Feeling: The Case of the Twitter API : Computational Culture.' In: *Computational Culture: A Journal of Software Studies* 3. URL: http://computationalculture.net/objects-of-intense-feeling-the-case-of-the-twitter-api/ (visited on 06/17/2019).

Bughin, Jacques, Eric Hazan, Eric Labaye, James Manyika, Peter Dahlström, Sree Ramaswamy, and C Cochin de Billy (2016). 'Digital Europe: Pushing the frontier, capturing the benefits.' In: *McKinsey Global Institute*.

Bughin, Jacques, Eric Hazan, Susan Lund, Peter Dahlström, Anna Wiesinger, and Amresh Subramaniam (2018). 'Skill shift: Automation and the future of the workforce.' In: *McKinsey Global Institute. McKinsey & Company*.

Burnett, Margaret M. and Brad A. Myers (2014). 'Future of End-user Software Engineering: Beyond the Silos.' In: *Proceedings of the on Future of Software Engineering*. FOSE 2014. Hyderabad, India: ACM, pp. 201–211. ISBN: 978-1-4503-2865-4. DOI: 10.1145/2593882.2593896. URL: http://doi.acm.org/10.1145/2593882.2593896.

Bush, Vannevar (1945a). 'As we may think.' In: *The Atlantic Monthly* 176.1, pp. 101–108.

Bush, Vannevar (1945b). 'As we may think.' In: *Atlantic Monthly* 176, pp. 101–108.

Bush, Vannevar et al. (1945). 'As we may think.' In: *The atlantic monthly* 176.1, pp. 101–108.

Campbell-Kelly, Martin (1995). 'Development and structure of the international software industry, 1950-1990.' In: *Business and economic history*, pp. 73–110.

Campbell-Kelly, Martin (2004). *From airline reservations to Sonic the Hedgehog: a history of the software industry*. MIT press.

Campbell-Kelly, Martin (2011). 'In praise of 'Wilkes, Wheeler, and Gill'.' In: *Communications of the ACM* 54.9, pp. 25–27.

Carretero, S, R Vuorikari, and Y Punie (2017). *DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use. Publications Office of the European Union EUR 28558 EN, DOI: 10.2760/38842*.

Castells, Manuel (2009). *The Rise of the Network Society*. 2nd ed. Vol. 1. The Information Age: Economy, Society, and Culture. Blackwell Publishers. ISBN: 978-0-631-22140-1.

Cate, Fred H (2010). 'The limits of notice and choice.' In: *IEEE Security & Privacy* 8.2, pp. 59–62.

Chang, Kerry Shih-Ping and Brad A. Myers (Apr. 2017). 'Gneiss.' In: *J. Vis. Lang. Comput.* 39.C, pp. 41–50. ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2016.07.004. URL: https://doi.org/10.1016/j.jvlc.2016.07.004.

Childs, Art (1976). 'Interfacial.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/.

Clifford, Damian, Inge Graef, and Peggy Valcke (2019). 'Pre-formulated Declarations of Data Subject Consent—Citizen-Consumer Empowerment and the Alignment of Data, Consumer and Competition Law Protections.' In: *German Law Journal* 20.5, pp. 679–721.

Coda (2019). URL: https://coda.io.

Cohen, Julie E (2019). *Between Truth and Power: The Legal Constructions of Informational Capitalism*. Oxford University Press, USA.

Commission nationale de l'informatique et des libertés (CNIL) (2019). *Délibération n° 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs) (rectificatif)*.

Conti, Gregory and Edward Sobiesk (2010). 'Malicious Interface Design: Exploiting the User.' In: *Proceedings of the 19th International Conference on World Wide Web*. ACM, pp. 271–280.

Conway, Jake R., Alexander Lex, and Nils Gehlenborg (2017). 'UpSetR: An R Package for the Visualization of Intersecting Sets and Their Properties.' In: *Bioinformatics* 33.18, pp. 2938–2940. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx364. URL: https://academic.oup.com/bioinformatics/article/33/18/2938/3884387 (visited on 09/19/2019).

Corporation, Lotus Development (1991). '@Functions and Macros Guide: Lotus 1-2-3 Release 2.3.' In:

Council of European Union (1990). *Directive on the minimum safety and health requirements for work with display screen equipment (90/270/EEC)*. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31990L0270.

Court of Justice of the European Union (2019a). *Case C-49/17* Fashion ID GmbH & Co.KG v Verbraucherzentrale NRW eV. *ECLI:EU:C:2019:629*.

Court of Justice of the European Union (2019b). *Case C-673/17* Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. *ECLI:EU:C:2019:801*.

Cranor, Lorrie (2002). *Web privacy with P3P*. Sebastopol, CA: O'Reilly Media.

Cranor, Lorrie Faith (2012). 'Necessary but Not Sufficient: Standardized Mechanisms for Privacy Notice and Choice The Economics of Privacy.' In: *Journal on Telecommunications and High Technology Law* 10.2, pp. 273–308.

Crémer, Jacques, Yves-Alexandre de Montjoye, Heike Schweitzer, European Commission, and Directorate-General for Competition (2019). *Competition policy for the digital era.* ISBN: 978-92-76-01946-6. URL: http://publications.europa.eu/publication/manifestation_identifier/PUB_KD0419345ENN.

Data Protection Commission (2020). 'Report by the Data Protection Commission on the use of cookies and other tracking technologies: Following a sweep conducted between August 2019 and December 2019.' In:

Datatilsynet (2020). *DMI's behandling af personoplysninger om hjemmesidebesøgende*. URL: https://www.datatilsynet.dk/tilsyn-og-afgoerelser/afgoerelser/2020/feb/dmis-behandling-af-personoplysninger-om-hjemmesidebesoegende.

Deb Fisher, Mark R. Warner (2019). 'Deceptive Experiences To Online Users Reduction (DETOUR) Act.' In: URL: https://www.scribd.com/document/405606873/Detour-Act-Final.

Degeling, Martin, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz (2018). 'We Value Your Privacy... Now Take Some Cookies: Measuring the GDPR's Impact on Web Privacy.' In: *arXiv preprint arXiv:1808.05096*.

Derwin, Doug (1987). 'Overheard...' In: *InfoWorld* 9.4.

diSessa, Andrea and Hal Abelson (Sept. 1986). 'Boxer: A Reconstructible Computational Medium.' In: *Commun. ACM* 29.9, pp. 859–868. ISSN: 0001-0782. DOI: 10.1145/6592.6595. URL: https://doi.org/10.1145/6592.6595.

diSessa, Andrea A. (2001). *Changing Minds: Computers, Learning, and Literacy*. Mit Press. ISBN: 9780262041805.

Dix, Alan (2007). 'Designing for appropriation.' In: *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 2*. British Computer Society, 27–30. URL: http://dl.acm.org/citation.cfm?id=1531415.

Docker (2019). URL: https://docker.com.

Doctorow, Cory (2019). 'Adversarial Interoperability.' In: *Electronic Frontier Foundation*. URL: https://www.eff.org/deeplinks/2019/10/adversarial-interoperability.

Douglas, Shawn M, Adam H Marblestone, Surat Teerapittayanon, Alejandro Vazquez, George M Church, and William M Shih (2009). 'Rapid prototyping of 3D DNA-origami shapes with caDNAno.' In: *Nucleic acids research* 37.15, pp. 5001–5006. DOI: 10.1093/nar/gkp436.

Dzieza, Josh (Dec. 19, 2018). 'Prime and Punishment: Dirty dealing in the $175 billion Amazon Marketplace.' In: *The Verge*. URL: https://www.theverge.com/2018/12/19/18140799/amazon-marketplace-scams-seller-court-appeal-reinstatement (visited on 06/21/2020).

Eagan, James R and John T Stasko (2008). 'The buzz: supporting user tailorability in awareness applications.' In: *Proceedings of the sigchi conference on human factors in computing systems*, pp. 1729–1738.

Edson, Joanna, John Greenstadt, Irwin Greenwald, Fletcher R. Jones, and Frank V. Wagner (1956). 'SHARE Reference Manual for the IBM 704.' In: URL: https://latimesblogs.latimes.com/thedailymirror/files/share59.pdf.

Ekbia, Hamid and Bonnie Nardi (2016). 'Social Inequality and HCI: The View from Political Economy.' In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. San Jose, California, USA: ACM, pp. 4997–5002. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858343. URL: http://doi.acm.org/10.1145/2858036.2858343.

Ellis, Clarence A and Simon J Gibbs (1989). 'Concurrency control in groupware systems.' In: *Acm Sigmod Record*. Vol. 18. 2. ACM, pp. 399–407.

End User Development (2006). Vol. 9. Human-Computer Interaction Series. Springer Netherlands. ISBN: 978-1-4020-4220-1. DOI: 10.1007/1-4020-5386-X. URL: http://link.springer.com/10.1007/1-4020-5386-X.

Energi-, Forsynings- og Klimaministeriet (2011). *Lov om elektroniske kommunikationsnet og -tjenester. LOV nr 169 af 03/03/2011*. URL: https://www.retsinformation.dk/eli/lta/2011/169.

Engelbart, Douglas C (1962). 'Augmenting human intellect: a conceptual framework.' In: *Summary Report, Stanford Research Institute, on Contract AF 49(638)-1024*.

Engelbart, Douglas C. (1988). 'Computer-supported Cooperative Work: A Book of Readings.' In: ed. by Irene Greif. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Chap. Toward High-performance Knowledge Workers (Reprint),

pp. 67–78. ISBN: 0-934613-57-5. URL: http://dl.acm.org/citation.cfm?id=49504.49507.

Engelbart, Douglas C. (1990). 'Knowledge-domain Interoperability and an Open Hyperdocument System.' In: *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*. CSCW '90. Los Angeles, California, USA: ACM, pp. 143–156. ISBN: 0-89791-402-3. DOI: 10.1145/99332.99351. URL: http://doi.acm.org/10.1145/99332.99351.

Engeström, Yrjö (2015). *Learning by Expanding*. Cambridge University Press.

Erhvervsministeriet (2011). *Bekendtgørelse om krav til information og samtykke ved lagring af eller adgang til oplysninger i slutbrugeres terminaludstyr*. URL: https://www.retsinformation.dk/eli/lta/2011/1148.

Erhvervsstyrelsen (2020). *Om Erhvervsstyrelsen*. URL: https://erhvervsstyrelsen.dk/om-erhvervsstyrelsen (visited on 07/06/2020).

Eurofound (2018). *Societal change and trust in institutions*. URL: https://www.mm.dk/misc/Democracy-Perception-Index-2018-1.pdf.

European Commission (2017). *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Communications)*.

European Commission (2020). *Digital Economy and Society Index (DESI) 2020 Denmark*.

European Commission and Directorate-General for Justice and Consumers and European Commission and Directorate-General Communication and Kantar (2019). *The General Data Protection Regulation: report.* ISBN: 978-92-76-08384-9. URL: https://data.europa.eu/doi/10.2838/579882.

European Commission and Directorate-General for Justice and Consumers and TNS Opinion Social (2015). *Data protection: report.* European Commission. ISBN: 978-92-79-48414-8. URL: http://dx.publications.europa.eu/10.2838/552336.

European Commission, Directorate-General for the Information Society and Media (2015). *ePrivacy directive, assessment of transposition, effectiveness and copmatibility with the proposed data protection regulation final report.* URL: http://bookshop.europa.eu/uri?target=EUB:NOTICE:KK0415268:EN:HTML.

European Data Protection Board (2019). 'First overview on the implementation of the GDPR and the roles and means of the national supervisory authorities.' In: URL: https://edpb.europa.eu/sites/edpb/files/files/file1/19_2019_edpb_written_report_to_libe_en.pdf.

European Data Protection Supervisor. *EDPS Opinion on the Proposal for a Regulation on Privacy and Electronic Communications (ePrivacy Regulation), Opinion 6/2017*. Brussels, BE: EDPS.

European Political Strategy Center (2016). 'The Future of Work: Skills and Resilience for a World of Change.' In: *Strategic Notes* 13.

European Union (1995). *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ 1995 L 281/31*.

European Union (2002). *Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications) OJ L 201*.

European Union (2009). 'Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services.' In: *Official Journal of the European Union*.

European Union (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1*.

European Union, Council of the (2020). *Interinstitutional File: 2017/0003(COD)*.

EUROSTAT (2008). *NACE rev. 2*. Office for Official Publications of the European Communities. ISBN: 978-92-79-04741-1.

Fakas, Georgios John, Anh Vu Nguyen, and Denis Gillet (2005). 'The Electronic Laboratory Journal: A Collaborative and Cooperative Learning Environment for Web-Based Experimentation.' In: *Computer Supported Cooperative Work (CSCW)* 14.3, pp. 189–216. ISSN: 1573-7551. DOI: 10.1007/s10606-005-3272-3. URL: https://doi.org/10.1007/s10606-005-3272-3.

Flanagan, John C (1954a). 'The critical incident technique.' In: *Psychological bulletin* 51.4, p. 327.

Flanagan, John C. (1954b). 'The Critical Incident Technique.' In: *Psychological bulletin* 51.4, p. 327. DOI: 10.1037/h0061470.

Flynn, Laurie (1989). 'Applications for DDE are Starting to Appear.' In: *InfoWorld* 11.44.

Fogg, Brian J (2009). 'A behavior model for persuasive design.' In: *Proceedings of the 4th international Conference on Persuasive Technology*. ACM, p. 40.

Forbrukerrådet (2019). *Deceived by Design: How tech companies use dark patterns to discourage us from exercising our rights to privacy*. URL: https://fil.forbrukerradet.no/wp-content/uploads/2018/06/2018-06-27-deceived-by-design-final.pdf.

Frankston, Robert M. (2015). 'Implementing VisiCalc.' In: URL: https://rmf.vc/implementingvisicalc.

Frich, Jonas, Lindsay MacDonald Vermeulen, Christian Remy, Michael Mose Biskjaer, and Peter Dalsgaard (2019). 'Mapping the landscape of creativity support tools in HCI.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–18.

Fuglseth, Anna Mette and Øystein Sørebø (2014). 'The effects of technostress within the context of employee use of ICT.' In: *Computers in Human Behavior* 40, 161–170. ISSN: 0747-5632. DOI: 10.1016/j.chb.2014.07.040.

Fuster, Gloria González (2014). *The emergence of personal data protection as a fundamental right of the EU*. Vol. 16. Springer Science & Business.

Galbraith, John Kenneth (1952). 'American Capitalism: The Concept of Countervailing Power.' In: *Boston: Houghton Mifflin*.

Gallie, W. B. (1955). 'Essentially contested concepts.' In: *Proceedings of the Aristotelian society*. Vol. 56. Wiley, pp. 167–198.

Gates, Bill et al. (1976). 'An open letter to hobbyists.' In: *Homebrew Computer Club Newsletter* 2.1, p. 2.

German Federal Ministry of Labour and Social Affairs (2015). 'Green Paper Work 4.0.' In: *Strategic Notes*.

Glöss, Mareike, Moira McGregor, and Barry Brown (2016). 'Designing for Labour: Uber and the On-Demand Mobile Workforce.' In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. San Jose, California, USA: ACM, pp. 1632–1643. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858476. URL: http://doi.acm.org/10.1145/2858036.2858476.

Goldberg, Adele (Oct. 1995). 'Why Smalltalk?' In: *Commun. ACM* 38.10, pp. 105–107. ISSN: 0001-0782. DOI: 10.1145/226239.226260. URL: http://doi.acm.org/10.1145/226239.226260.

Goodman, D (1988). *The complete Hyper Card Handbook. Bantam Books*.

Goransson, Bengt, Mats Lind, Else Pettersson, Bengt Sandblad, and Patrik Schwalbe (1987). 'The Interface is Often Not the Problem.' In: *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*. CHI '87. Toronto, Ontario, Canada: ACM, pp. 133–136. ISBN: 0-89791-213-6. DOI: 10.1145/29933.30872. URL: http://doi.acm.org/10.1145/29933.30872.

Gray, Colin M, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L Toombs (2018). 'The dark (patterns) side of UX design.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 534.

Green, Francis (2013). *Skills and skilled work: an economic and social analysis*. Oxford University Press.

Greenhalgh, Trisha, Henry WW Potts, Geoff Wong, Pippa Bark, and Deborah Swinglehurst (2009). 'Tensions and paradoxes in electronic patient record research: A systematic literature review using the meta-narrative method.' In: *The Milbank Quarterly* 87.4, pp. 729–788.

Grosse, Meghan (2020). 'Laying the foundation for a commercialized internet: international internet governance in the 1990s.' In: *Internet Histories* 0.0, pp. 1–16. DOI: 10.1080/24701475.2020.1769890. eprint: https://doi.org/10.1080/24701475.2020.1769890. URL: https://doi.org/10.1080/24701475.2020.1769890.

Grossman, Gene M, Esteban Rossi-Hansberg, et al. (2006). 'The rise of offshoring: it's not wine for cloth anymore.' In: *The new economic geography: effects and policy implications*, pp. 59–102.

Grudin, Jonathan (1995). 'Groupware and Social Dynamics: Eight Challenges for Developers.' In: *Readings in Human–Computer Interaction*. Elsevier, pp. 762–774. DOI: 10.1016/B978-0-08-051574-8.50079-0.

Guendert, Steve (2011). 'Mainframe History and the First User's Groups (SHARE).' In: p. 2. URL: https://www.cmg.org/wp-content/uploads/2011/05/m_79_5.pdf.

Guo, Philip J. and Dawson Engler (2011). 'CDE: Using System Call Interposition to Automatically Create Portable Software Packages.' In: *USENIXATC'11 Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, p. 21. URL: https://dl.acm.org/citation.cfm?id=2002202.

Gutwin, Carl and Saul Greenberg (2002). 'A Descriptive Framework of Workspace Awareness for Real-Time Groupware.' In: *Computer Supported Cooperative Work (CSCW)* 11.3-4, pp. 411–446. DOI: 10.1023/A:1021271517844.

Haigh, Thomas (2002). 'Software in the 1960s as Concept, Service, and Product.' In: *IEEE Annals of the History of Computing* 24.1, pp. 5–13.

Handel, Mark J. and Steven Poltrock (2011). 'Working Around Official Applications: Experiences from a Large Engineering Project.' In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. CSCW '11. Hangzhou, China: ACM, pp. 309–312. ISBN: 978-1-4503-0556-3. DOI: 10.1145/1958824.1958870. URL: http://doi.acm.org/10.1145/1958824.1958870.

Harper, R. R., J. A. Hughes, and D. Z. Shapiro (1990). 'Harmonious Working and CSCW: Computer Technology and Air Traffic Control.' In: *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*. NLD: North-Holland Publishing Co., 225–234. ISBN: 044488811X.

Harper, Richard and Abigail Sellen (1995). 'Collaborative Tools and the Practicalities of Professional Work at the International Monetary Fund.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., pp. 122–129. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223920. URL: http://dx.doi.org/10.1145/223904.223920.

Harrison, Andrew, Ian Harvey, Andrew Jones, David Rogers, and Ian Taylor (2011). 'Object Reuse and Exchange for Publishing and Sharing Workflows.' In: *Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science*. WORKS '11. Seattle, Washington, USA: Association for Computing Machinery, 67–76. ISBN: 9781450311007. DOI: 10.1145/2110497.2110506. URL: https://doi.org/10.1145/2110497.2110506.

Harvey, David (2007). 'Neoliberalism as creative destruction.' In: *The annals of the American academy of political and social science* 610.1, pp. 21–44.

Hathaway, Terry. 'Neoliberalism as Corporate Power.' In: *Competition & Change* ().

Head, Andrew, Fred Hohman, Titus Barik, Steven M. Drucker, and Robert DeLine (2019). 'Managing Messes in Computational Notebooks.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, p. 270. DOI: 10.1145/3290605.3300500.

Holdgraf, Chris, Aaron Culich, Ariel Rokem, Fatma Deniz, Maryana Alegro, and Dani Ushizima (2017). 'Portable Learning Environments for Hands-On Computational Instruction: Using Container- and Cloud-Based Technology to Teach Data Science.' In: *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. PEARC17. New Orleans, LA, USA: Association for Computing Machinery. ISBN: 9781450352727. DOI: 10.1145/3093338.3093370. URL: https://doi.org/10.1145/3093338.3093370.

Holtman, Koen (Aug. 11, 1995). *Non-persistent Cookie proposal*. www-talk electronic mailing list message. URL: https://lists.w3.org/Archives/Public/www-talk/msg01499.html.

(IDA), Ingeniørforeningen (2020). *Shoshana Zuboff meets Margrethe Vestager: A conversation about a future digital Europe - webinar*. URL: https://ida.dk/

arrangementer-og-kurser/arrangementer/shoshana-zuboff-meets-margrethe-vestager-a-conversation-about-a-future-digital-europe-webinar-336843 (visited on 07/10/2020).

Information Commissioner's Office (July 2019a). *Guidance on the Use of Cookies and Similar Technologies*. Wilmslow, Cheshire: ICO.

Information Commissioner's Office (June 2019b). *Update Report into Adtech and Real Time Bidding*. Wilmslow, Cheshire: ICO.

Ingalls, Dan, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay (Oct. 1997). 'Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself.' In: *SIGPLAN Not.* 32.10, pp. 318–326. ISSN: 0362-1340. DOI: 10.1145/263700.263754. URL: http://doi.acm.org/10.1145/263700.263754.

International Data Systems, Inc. (1976). 'Patchable Star Trek/Space War Program Offered.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/.

Irani, Lilly C. and M. Six Silberman (2013). 'Turkopticon: Interrupting Worker Invisibility in Amazon Mechanical Turk.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, pp. 611–620. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470742. URL: http://doi.acm.org/10.1145/2470654.2470742.

ISO/IEC 2382-01 (1993). *Information Technology Vocabulary, Fundamental Terms*.

Jaimovich, Nir and Henry E Siu (2012). 'Job Polarization and Jobless Recoveries.' In: *National Bureau of Economic Research*. DOI: 10.3386/w18334. URL: http://www.nber.org/papers/w18334.

Jensen, Carlos and Colin Potts (2004). 'Privacy policies as decision-making tools: an evaluation of online privacy notices.' In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, pp. 471–478.

Jirotka, Marina, Charlotte P. Lee, and Gary M. Olson (Aug. 2013). 'Supporting Scientific Collaboration: Methods, Tools and Concepts.' In: *Comput. Supported Coop. Work* 22.4-6, pp. 667–715. ISSN: 0925-9724. DOI: 10.1007/s10606-012-9184-0. URL: http://dx.doi.org/10.1007/s10606-012-9184-0.

Johnson, Jeff, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey (Sept. 1989). 'The Xerox Star: A Retrospective.' In: *Computer* 22.9, pp. 11–26, 28–29. ISSN: 0018-9162. DOI: 10.1109/2.35211. URL: http://dx.doi.org/10.1109/2.35211.

Johnson, Luanne (2002). 'Creating the software industry-recollections of software company founders of the 1960s.' In: *IEEE Annals of the History of Computing* 24.1, pp. 14–42.

Jones, Meg Leta (2020). 'Surveillance Capitalism Online: Cookies, Notice Choice, and Web Privacy.' In: *Surveillance Capitalism in America: From Slavery to Social Media*. Ed. by Josh Lauer and Kenneth Lipartito. University of Pennsylvania Press, p. 23.

Juypter Notebook (2019). URL: https://jupyter.org.

Kamara, Irene and Eleni Kosta (2016). 'Do Not Track Initiatives: Regaining the Lost User Control.' In: *International Data Privacy Law* 6.4, pp. 276–290. ISSN: 2044-3994. DOI: 10/gdxwds. (Visited on 12/28/2018).

Kaptelinin, Victor and Liam J. Bannon (2012). 'Interaction Design Beyond the Product: Creating Technology-Enhanced Activity Spaces.' In: *Human–Computer Interaction* 27.3, pp. 277–309. DOI: 10.1080/07370024.2011.646930. eprint: http:

//www.tandfonline.com/doi/pdf/10.1080/07370024.2011.646930. URL: http://www.tandfonline.com/doi/abs/10.1080/07370024.2011.646930.

Karger, David (2007). 'Haystack: Per-User Information Environments Based on Semistructured Data.' In: *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*. Ed. by Victor Kaptelinin and Mary Czerwinski. Cambridge, MA, USA: MIT Press. Chap. 3, pp. 49–100.

Kay, Alan (2007). 'The real computer revolution hasn't happened yet.' In: *Viewpoints Research Institute* 15.

Kay, Alan and Adele Goldberg (1977). 'Personal dynamic media.' In: *Computer* 10.3, pp. 31–41.

Kay, Alan C. (1972). 'A Personal Computer for Children of All Ages.' In: *Proceedings of the ACM Annual Conference - Volume 1*. ACM '72. Boston, Massachusetts, USA: Association for Computing Machinery. ISBN: 9781450374910. DOI: 10.1145/800193.1971922. URL: https://doi.org/10.1145/800193.1971922.

Kelley, Patrick Gage, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder (2009). 'A nutrition label for privacy.' In: *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, p. 4.

Kery, Mary Beth and Brad A. Myers (2018). 'Interactions for Untangling Messy History in a Computational Notebook.' In: *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, pp. 147–155. DOI: 10.1109/VLHCC.2018.8506576.

Kery, Mary Beth, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers (2018). 'The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 174. DOI: 10.1145/3173574.3173748.

Kierkegaard, Sylvia Mercado (2005). 'How the cookies (almost) crumbled: Privacy & lobbyism.' In: *Computer Law & Security Review* 21.4, pp. 310–322.

Kiestadt, Ralph (1976). 'Large Scale Systems: Software Choices for Star Trek.' In: *SCCS INTERFACE* June. URL: https://archive.org/details/sccs_v1n7/.

Kitchin, Rob (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage.

Kjær, Jakob Sorgenfri (2020). *Bombe under hjemmesider: DMI får alvorlig kritik i banebrydende afgørelse*. URL: https://politiken.dk/viden/Tech/art7657203/DMI-f\%C3\%A5r-alvorlig-kritik-i-banebrydende-afg\%C3\%B8relse.

Klokmose, Clemens N., James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon (2015a). 'Webstrates: Shareable Dynamic Media.' In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. Charlotte, NC, USA: ACM, pp. 280–290. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807446. URL: http://doi.acm.org/10.1145/2807442.2807446.

Klokmose, Clemens N, James R Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon (2015b). 'Webstrates: shareable dynamic media.' In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, pp. 280–290.

Klokmose, Clemens N. and Pär-Ola Zander (2010). 'Rethinking Laboratory Notebooks.' In: *Proceedings of COOP 2010*. Springer, pp. 119–139. DOI: 10.1007/978-1-84996-211-7_8.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, et al. (2016). 'Jupyter Notebooks–a publishing format for reproducible computational workflows.' In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87–90.

Knuth, Donald E. (May 1984). 'Literate Programming.' In: *Comput. J.* 27.2, pp. 97–111. ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. URL: http://dx.doi.org/10.1093/comjnl/27.2.97.

Korsgaard, Henrik, Clemens Nylandsted Klokmose, and Susanne Bødker (2016). 'Computational Alternatives in Participatory Design: Putting the T Back in Sociotechnical Research.' In: *Proceedings of the 14th Participatory Design Conference: Full Papers - Volume 1*. PDC '16. Aarhus, Denmark: ACM, pp. 71–79. ISBN: 978-1-4503-4046-5. DOI: 10.1145/2940299.2940314. URL: http://doi.acm.org/10.1145/2940299.2940314.

Korzeniowksi, Paul (1984). 'Multi-application Packages: Who Needs Them?' In: *InfoWorld* 18.33.

Kosta, Eleni (2013a). 'Peeking into the cookie jar: the European approach towards the regulation of cookies.' In: *International journal of law and information technology* 21.4, pp. 380–406.

Kosta, Eleni (2013b). 'Peeking into the Cookie Jar: The European Approach towards the Regulation of Cookies.' In: *International Journal of Law and Information Technology* 21.4, pp. 380–406. DOI: 10.1093/ijlit/eat011.

Krahn, Robert, Dan Ingalls, Robert Hirschfeld, Jens Lincke, and Krzysztof Palacz (2009). 'Lively Wiki a Development Environment for Creating and Sharing Active Web Content.' In: *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*. WikiSym '09. Orlando, Florida: ACM, 9:1–9:10. ISBN: 978-1-60558-730-1. DOI: 10.1145/1641309.1641324. URL: http://doi.acm.org/10.1145/1641309.1641324.

Labguru (2019). URL: https://labguru.com.

Lampinen, Airi and Barry Brown (2017). 'Market Design for HCI: Successes and Failures of Peer-to-Peer Exchange Platforms.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, pp. 4331–4343. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025515. URL: http://doi.acm.org/10.1145/3025453.3025515.

Lasswell, Harold D. (1936). *Politics: Who Gets What, When, How*. Whittlesey House.

Lazar, Jonathan et al. (May 2016). 'Human–Computer Interaction and International Public Policymaking: A Framework for Understanding and Taking Future Actions.' In: *Found. Trends Hum.-Comput. Interact.* 9.2, 69–149. ISSN: 1551-3955. DOI: 10.1561/1100000062. URL: https://doi.org/10.1561/1100000062.

Leenes, Ronald and Eleni Kosta (2015). 'Taming the cookie monster with dutch law–a tale of regulatory failure.' In: *Computer Law & Security Review* 31.3, pp. 317–335.

Lessig, Lawrence (1999). *Code and other laws of cyberspace*. Basic Books. ISBN: 978-0-465-03912-8.

Lewicki, Roy J., Bruce Barry, and David M. Saunders (2016). *Essentials of Negotiation*. 6th ed. McGraw Hill. ISBN: 978-0-07-7862466.

Lex, A., N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister (2014). 'UpSet: Visualization of Intersecting Sets.' In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 1983–1992. DOI: 10.1109/TVCG.2014.2346248.

Licklider, J. C. R. (1960). 'Man-Computer Symbiosis.' In: *IRE Transactions on Human Factors in Electronics* HFE-1, pp. 4–11.

Lorrie Faith Cranor, Joseph Reagle Jr. (1997). 'Designing a Social Protocol: Lessons Learned from the Platform for Privacy Preferences.' In: *Proceedings of the Telecommunications Policy Research Conference*.

Mace, Scott (1983). 'Software accessories enhance software programs.' In: *InfoWorld* 5.10.

Mackay, Wendy (1990). 'Users and customizable software: A co-adaptive phenomenon.' PhD thesis. Massachusetts Institute of Technology.

Mackay, Wendy E. (2003). 'The Missing Link: Integrating Paper and Electronic Documents.' In: *Proceedings of the 15th Conference on L'Interaction Homme-Machine*. IHM '03. Caen, France: ACM, pp. 1–8. ISBN: 1-58113-803-2. DOI: 10.1145/1063669.1063671. URL: http://doi.acm.org/10.1145/1063669.1063671.

MacLean, Allan, Kathleen Carter, Lennart Lövstrand, and Thomas Moran (1990). 'User-tailorable systems: pressing the issues with buttons.' In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 175–182.

Mahieu, Rene, Joris van Hoboken, and Hadi Asghari (2019). 'Responsibility for Data Protection in a Networked World: On the Question of the Controller, Effective and Complete Protection and Its Application to Data Access Rights in Europe.' In: *Journal of Intellectual Property, Information Technology and Electronic Commerce Law* 10.1, pp. 84–104. (Visited on 09/03/2019).

Maloney, John H and Randall B Smith (1995). 'Directness and liveness in the morphic user interface construction environment.' In: *Proceedings of the 8th annual ACM symposium on User interface and software technology*. ACM, pp. 21–28.

Mandl, Irene, Maurizio Curtarelli, Sara Riso, Oscar Vargas, and Elias Gerogiannis (2015a). *New forms of employment*. Vol. 2. Publications Office of the European Union Eurofond, Luxembourg.

Mandl, Irene, Maurizio Curtarelli, Sara Riso, Oscar Vargas, and Elias Gerogiannis (2015b). *New forms of employment*. Vol. 2. Publications Office of the European Union.

Manners, Ian (2002). 'Normative power Europe: a contradiction in terms?' In: *JCMS: Journal of common market studies* 40.2, pp. 235–258.

Martin, David, Benjamin V. Hanrahan, Jacki O'Neill, and Neha Gupta (2014). 'Being a Turker.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: ACM, pp. 224–235. ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531663. URL: http://doi.acm.org/10.1145/2531602.2531663.

Massey, Charlotte, Thomas Lennig, and Steve Whittaker (2014). 'Cloudy Forecast: An Exploration of the Factors Underlying Shared Repository Use.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, pp. 2461–2470. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557042. URL: http://doi.acm.org/10.1145/2556288.2557042.

Mathur, Arunesh, Gunes Acar, Michael J Friedman, Elena Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan (2019). 'Dark patterns at scale: Findings from a crawl of 11K shopping websites.' In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW, p. 81.

Matsaganis, Manos, Erhan Özdemir, Terry Ward, and Alkistis Zavakou (2016). *Non-Standard Employment and Access to Social Security Benefits*. Directorate-General for Employment, Social Affairs and Inclusion, European Commission.

Matte, Célestin, Nataliia Bielova, and Cristiana Santos (2019a). 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.' In: Under submission. URL: https://arxiv.org/abs/1911.09964v1.

Matte, Célestin, Nataliia Bielova, and Cristiana Santos (2019b). 'Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework.' In: *Under review*. URL: arXiv: 1911.09964.

McCarthy, John (2019). *Over 90% of users consent to GDPR requests says Quantcast after enabling 1bn of them*. https://www.thedrum.com/news/2018/07/31/over-90-users-consent-gdpr-requests-says-quantcast-after-enabling-1bn-them.

McDonald, A. M. and L. F. Cranor (2008). 'The cost of reading privacy policies.' In: *I/S: A Journal of Law and Policy for the Information Society* 4, pp. 540 –565.

McGeever, Christine (1984). 'A Look at Lotus for the Mac.' In: *InfoWorld* 6.47.

Mejias, Ulises A and Nick Couldry (2019). 'Datafication.' In: *Internet Policy Review* 8.4.

Metropolis, Nick and Jack Worlton (1980). 'A Trilogy on Errors in the History of Computing.' In: *Annals of the History of Computing* 2.1, pp. 49–59.

Miller, Michael J. (1994). 'Are They Suites Yet.' In: *PC Magazine* 13.18.

Millett, Lynette I, Batya Friedman, and Edward Felten (2001). 'Cookies and web browser design: Toward realizing informed consent online.' In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 46–52.

Millman, K Jarrod and Fernando Pérez (2014). 'Developing open-source scientific practice.' In: *Implementing Reproducible Research* 149.

Ministry of Science, Innovation and Higher Education (2013). *Ministerial Order on the PhD Degree Programme at the Universities and Certain Higher Artistic Educational Institutions*. https://www.retsinformation.dk/eli/lta/2013/1039.

Montulli, Lou (Apr. 18, 1995). *Re: Session tracking*. www-talk electronic mailing list message. URL: https://lists.w3.org/Archives/Public/www-talk/1995MarApr/0462.html.

Mørch, Anders (1997). 'Three levels of end-user tailoring: Customization, integration, and extension.' In: *Computers and design in context*. Ed. by Morten Kyng and Lars Mathiassen. MIT Press, pp. 51–76.

Morozov, Evgeny (2013). *To save everything, click here: The folly of technological solutionism*. Public Affairs.

Muller, Michael, David R. Millen, and Jonathan Feinberg (2010). 'Patterns of Usage in an Enterprise File-sharing Service: Publicizing, Discovering, and Telling the News.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, pp. 763–766. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753438. URL: http://doi.acm.org/10.1145/1753326.1753438.

Myers, Brad, Scott E. Hudson, and Randy Pausch (Mar. 2000). 'Past, Present, and Future of User Interface Software Tools.' In: *ACM Trans. Comput.-Hum. Interact.* 7.1, pp. 3–28. ISSN: 1073-0516. DOI: 10.1145/344949.344959. URL: http://doi.acm.org/10.1145/344949.344959.

n.a. (1975). 'Building your own computer won't be a piece of cake.' In: *Popular Electronics* 4.

Nichols, David A., Pavel Curtis, Michael Dixon, and John Lamping (1995). 'High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System.' In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. Pittsburgh, Pennsylvania, USA: ACM, pp. 111–120. ISBN: 0-89791-709-X. DOI: 10.1145/215585.215706. URL: http://doi.acm.org/10.1145/215585.215706.

Nissenbaum, H. (2011). 'A contextual approach to privacy online.' In: *Daedalus* 140.4, pp. 32–48.

Nooney, Laine, Kevin Driscoll, and Kera Allen (2020). 'From Programming to Products: Softalk Magazine and the Rise of the Personal Computer User.' In: *Information & Culture* 55.2, pp. 105–129.

Norberg, Arthur (1983). 'An Interview with Walter Bauer.' In: *Charles Babbage Institute*. URL: https://conservancy.umn.edu/bitstream/handle/11299/107108/oh061wb.pdf.

Norman, Donald A. (1998). *The Invisible Computer*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-14065-9.

Notion (2019). URL: https://notion.so.

Nouwens, Midas, Marcel Borowski, Bjarke Fog, and Clemens Nylandsted Klokmose (2020a). 'Between Scripts and Applications: Computational Media for the Frontier of Nanoscience.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376287. URL: https://doi.org/10.1145/3313831.3376287.

Nouwens, Midas and Clemens Nylandsted Klokmose (2018). 'The Application and Its Consequences for Non-Standard Knowledge Work.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173973. URL: https://doi.org/10.1145/3173574.3173973.

Nouwens, Midas, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal (2020b). 'Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence.' In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Com-

puting Machinery, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376321. URL: https://doi.org/10.1145/3313831.3376321.

Nouwens, Midas, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal (2020c). 'Dark Patterns Post-GDPR: Scraping Consent Interface Designs and Demonstrating their Influence.' In: *Conditionally accepted for CHI 2020*. ACM.

Obar, Jonathan A. and Anne Oeldorf-Hirsch (2018). 'The biggest lie on the Internet: ignoring the privacy policies and terms of service policies of social networking services.' In: *Information, Communication & Society* 0.0, pp. 1–20. DOI: 10.1080/1369118X.2018.1486870. eprint: https://doi.org/10.1080/1369118X.2018.1486870. URL: https://doi.org/10.1080/1369118X.2018.1486870.

Observable (2019). URL: https://observablehq.com.

OECD (2015). *In It Together: Why Less Inequality Benefits All*. Paris: OECD Publishing. DOI: http://dx.doi.org/10.1787/9789264235120-en. URL: /content/book/9789264235120-en.

Oleksik, Gerard, Natasa Milic-Frayling, and Rachel Jones (2012). 'Beyond Data Sharing: Artifact Ecology of a Collaborative Nanophotonics Research Centre.' In: *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*. CSCW '12. Seattle, Washington, USA: Association for Computing Machinery, 1165–1174. ISBN: 9781450310864. DOI: 10.1145/2145204.2145376. URL: https://doi.org/10.1145/2145204.2145376.

Oleksik, Gerard, Natasa Milic-Frayling, and Rachel Jones (2014). 'Study of Electronic Lab Notebook Design and Practices That Emerged in a Collaborative Scientific Environment.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: Association for Computing Machinery, 120–133. ISBN: 9781450325400. DOI: 10.1145/2531602.2531709. URL: https://doi.org/10.1145/2531602.2531709.

Olsen Jr., Dan R. (2007). 'Evaluating User Interface Systems Research.' In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, pp. 251–258. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294256. URL: http://doi.acm.org/10.1145/1294211.1294256.

Party, Article 29 Working (2009). *OPINION 1/2009 on the proposals amending Directive 2002/58 on privacy and electronic communications (e-Privacy Directive), WP159*. European Commission.

Party, Article 29 Working (2017). *Opinion 01/2017 on the Proposed Regulation for the ePrivacy Regulation (2002/58/EC)*. European Commission.

Pasquale, Frank (2016). 'Two Narratives of Platform Capitalism.' In: *Yale Law & Policy Review* 35, pp. 309–320.

Pérez, Fernando and Brian Granger (2015). *Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science*. URL: https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58.

Perkel, Dan (2008). 'Copy and Paste Literacy: Literacy practices in the production of a MySpace profile.' English. In: *Informal Learning and Digital Media*. Ed. by Kirsten Drotner, Siggaard Jensen, Hans, Schrøder, and Kim Christian. Cambridge Scholars Press. Chap. 10, pp. 203–224.

Piersol, Kurt (1994). 'Under the Hood: A Close-Up of OpenDoc.' In: *BYTE* 19, pp. 183–188. URL: http://archive.li/zPfWW.

Pike, Rob and Brian W Kernighan (1984). 'The UNIX System: Program Design in the UNIX Environment.' In: *AT&T Bell Laboratories Technical Journal* 63.8, pp. 1595–1605.

Plank, Thomas, Hans-Christian Jetter, Roman Rädle, Clemens N. Klokmose, Thomas Luger, and Harald Reiterer (2017). 'Is Two Enough?! Studying Benefits, Barriers, and Biases of Multi-Tablet Use for Collaborative Visualization.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. DOI: 10.1145/3025453.3025537.

Rädle, Roman, Midas Nouwens, Kristian Antonsen, James R Eagan, and Clemens N Klokmose (2017). 'Codestrates: Literate computing with webstrates.' In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 715–725.

Rädle, Roman, Midas Nouwens, Kristian Antonsen, James R. Eagan, and Clemens N. Klokmose (2017). 'Codestrates: Literate Computing with Webstrates.' In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST '17. Québec City, QC, Canada: ACM, pp. 715–725. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126642. URL: http://doi.acm.org/10.1145/3126594.3126642.

Raskin, Jef (2000). *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional.

Reeder, Robert W, Lujo Bauer, Lorrie Faith Cranor, Michael K Reiter, Kelli Bacon, Keisha How, and Heather Strong (2008). 'Expandable grids for visualizing and authoring computer security policies.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1473–1482.

Rosch, Winn L. (1985). 'Can Integrated Software Co-Exist with Windows?' In: *PC Magazine* 4.2.

Rosenberger, Robert (2009). 'The sudden experience of the computer.' In: *Ai & Society* 24.2, pp. 173–180.

Roubert, Francois and Mark Perry (2013). 'Putting the Lab in the Lab Book: Supporting Coordination in Large, Multi-site Research.' In: *HCI 2013 - 27th International British Computer Society Human Computer Interaction Conference: The Internet of Things*. URL: http://dl.acm.org/citation.cfm?id=2578048.2578066.

Rule, Adam, Ian Drosos, Aurélien Tabard, and James D. Hollan (2018). 'Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding.' In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW, p. 150. DOI: 10.1145/3274419.

Rule, Adam, Aurélien Tabard, and James D. Hollan (2018). 'Exploration and Explanation in Computational Notebooks.' In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, p. 32. DOI: 10.1145/3173574.3173606.

Salovaara, Antti, Antti Oulasvirta, and Giulio Jacucci (2017). 'Evaluation of Prototypes and the Problem of Possible Futures.' In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado,

USA: ACM, pp. 2064–2077. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025658. URL: http://doi.acm.org/10.1145/3025453.3025658.

Sanchez-Rola, Iskander, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos (2019). 'Can I Opt Out Yet?: GDPR and the Global Illusion of Cookie Control.' In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS '19. Auckland, New Zealand: ACM, pp. 340–351. ISBN: 978-1-4503-6752-3. DOI: 10.1145/3321705.3329806. URL: http://doi.acm.org/10.1145/3321705.3329806.

Satyanarayan, Arvind, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer (2017). 'Vega-lite: A grammar of interactive graphics.' In: *IEEE Transactions on Visualization and Computer Graphics* 23.1, pp. 341–350.

Scannell, Ed (1984). 'IBM User Groups.' In: *Computerworld* 8.October.

Schaub, Florian, Rebecca Balebako, Adam L Durity, and Lorrie Faith Cranor (2015). 'A design space for effective privacy notices.' In: *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pp. 1–17.

Schmidt, Kjeld and Ina Wagner (2004). 'Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning.' In: *Computer Supported Cooperative Work (CSCW)* 13.5, pp. 349–408. ISSN: 1573-7551. DOI: 10.1007/s10606-004-5059-3. URL: https://doi.org/10.1007/s10606-004-5059-3.

Schulte, Eric and Dan Davison (2011). 'Active documents with org-mode.' In: *Computing in Science & Engineering* 13.3, pp. 66–73.

Sellen, Abigail J. and Richard H.R. Harper (2003). *The Myth of the Paperless Office*. Cambridge, MA, USA: MIT Press. ISBN: 026269283X.

Sellen, Abigail J., Rachel Murphy, and Kate L. Shaw (2002). 'How Knowledge Workers Use the Web.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, pp. 227–234. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503418. URL: http://doi.acm.org/10.1145/503376.503418.

SHARE Program Library Agency (1977). 'User's Guide and Catalog of Programs.' In: URL: http://www.bitsavers.org/pdf/ibm/share/SHARE_PgmCatalog_Jan77.pdf.

Shrayer, Michael (1977). *The Electric Pencil Word Processor: Operator's Manual*.

Singer, Natasha (May 2016). 'When Websites Won't Take No for an Answer.' In: *New York Times*. URL: https://www.nytimes.com/2016/05/15/technology/personaltech/when-websites-wont-take-no-for-an-answer.html?mcubz=0&_r=0.

Skidmore, Jenifer L., Matthew J. Sottile, Janice E. Cuny, and Allen D. Malony (1998). 'A Prototype Notebook-based Environment for Computational Tools.' In: *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*. SC '98. San Jose, CA: IEEE Computer Society, pp. 1–15. ISBN: 0-89791-984-X. URL: http://dl.acm.org/citation.cfm?id=509058.509080.

Smith, Spencer (2018). 'Beyond Software Carpentry.' In: *Proceedings of the International Workshop on Software Engineering for Science*. SE4Science '18. Gothenburg, Sweden: Association for Computing Machinery, 32–39. ISBN: 9781450357487. DOI: 10.1145/3194747.3194749. URL: https://doi.org/10.1145/3194747.3194749.

Solon, Olivia (Jan. 11, 2018). 'Uber developed secret system to lock down staff computers in a police raid.' In: *The Guardian*. URL: https://www.theguardian.com/

technology/2018/jan/11/uber-developed-secret-system-to-lock-down-staff-computers-in-a-police-raid (visited on 06/21/2020).

Sørensen, Jannick and Sokol Kosta (2019). 'Before and After GDPR: The Changes in Third Party Presence at Public and Private European Websites.' In: *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: ACM, pp. 1590–1600. ISBN: 978-1-4503-6674-8. DOI: 10.1145/3308558.3313524. URL: http://doi.acm.org/10.1145/3308558.3313524.

Spaa, Anne, Abigail Durrant, Chris Elsden, and John Vines (2019). 'Understanding the Boundaries between Policymaking and HCI.' In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–15.

Su, Norman Makoto and Gloria Mark (2008). 'Designing for Nomadic Work.' In: *Proceedings of the 7th ACM Conference on Designing Interactive Systems*. DIS '08. Cape Town, South Africa: ACM, pp. 305–314. ISBN: 978-1-60558-002-9. DOI: 10.1145/1394445.1394478. URL: http://doi.acm.org/10.1145/1394445.1394478.

Supervisor, European Data Protection (2018). 'EDPS Opinion on the legislative package "A New Deal for Consumers".' In: URL: https://edps.europa.eu/sites/edp/files/publication/18-10-05_opinion_consumer_law_en.pdf.

Tabard, Aurélien, Juan-David Hincapié-Ramos, Morten Esbensen, and Jakob E. Bardram (2011). 'The eLabBench: An Interactive Tabletop System for the Biology Laboratory.' In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS '11. Kobe, Japan: ACM, pp. 202–211. ISBN: 978-1-4503-0871-7. DOI: 10.1145/2076354.2076391. URL: http://doi.acm.org/10.1145/2076354.2076391.

Taivalsaari, Antero, Tommi Mikkonen, Dan Ingalls, and Krzysztof Palacz (2008). *Web Browser As an Application Platform: The Lively Kernel Experience*. Mountain View, CA, USA.

Talbott, Tara, Michael Peterson, Jens Schwidder, and James D. Myers (2005). 'Adapting the Electronic Laboratory Notebook for the Semantic Era.' In: *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005.* Pp. 136–143. DOI: 10.1109/ISCST.2005.1553305.

Tanimoto, Steven L (2013). 'A perspective on the evolution of live programming.' In: *2013 1st International Workshop on Live Programming (LIVE)*. IEEE, pp. 31–34.

Tarafdar, Monideepa, Cary L. Cooper, and Jean-François Stich (2019). 'The technostress trifecta - techno eustress, techno distress and design: Theoretical directions and an agenda for research.' In: *Information Systems Journal* 29.1, 6–42. ISSN: 1365-2575. DOI: 10.1111/isj.12169.

Taylor, Linnet (2020). 'Public actors without public values: legitimacy, domination and the regulation of the technology sector.' In:

Tchounikine, Pierre (2017). 'Designing for Appropriation: A Theoretical Account.' In: *Human–Computer Interaction* 32.4, 155–195. ISSN: 0737-0024, 1532-7051. DOI: 10.1080/07370024.2016.1203263.

Teodoro, Rannie, Pinar Ozturk, Mor Naaman, Winter Mason, and Janne Lindqvist (2014). 'The Motivations and Experiences of the On-demand Mobile Workforce.' In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '14. Baltimore, Maryland, USA: ACM,

pp. 236–247. ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531680. URL: http://doi.acm.org/10.1145/2531602.2531680.

Thaler, Richard H and Cass R Sunstein (2009). *Nudge: Improving decisions about health, wealth, and happiness*. Penguin.

Tobin, Oisin (2019). 'Cookie consent revisited.' In: *Privacy and Data Protection* 19 (5), p. 11.

Tomlinson, Bill et al. (2012). 'Massively Distributed Authorship of Academic Papers.' In: *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '12. Austin, Texas, USA: ACM, pp. 11–20. ISBN: 978-1-4503-1016-1. DOI: 10.1145/2212776.2212779. URL: http://doi.acm.org/10.1145/2212776.2212779.

Trevisan, Martino, Stefano Traverso, Eleonora Bassi, and Marco Mellia (2019). '4 Years of EU Cookie Law: Results and Lessons Learned.' In: *Proceedings on Privacy Enhancing Technologies* 2019.2, pp. 126–145.

Trigg, Randall H., Thomas P. Moran, and Frank G. Halasz (1987a). 'Adaptability and Tailorability in NoteCards.' In: *Human–Computer Interaction–INTERACT '87*. Elsevier, 723–728. ISBN: 978-0-444-70304-0. DOI: 10.1016/B978-0-444-70304-0.50117-5. URL: http://linkinghub.elsevier.com/retrieve/pii/B9780444703040501175.

Trigg, Randall H, Thomas P Moran, and Frank G Halasz (1987b). 'Adaptability and tailorability in NoteCards.' In: *Human–Computer Interaction–INTERACT'87*. Elsevier, pp. 723–728.

Tufekci, Zeynep (2017). *Twitter and tear gas: The power and fragility of networked protest*. Yale University Press.

Turner, Phil and Susan Turner (1997). 'Supporting Cooperative Working Using Shared Notebooks.' In: *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*. Dordrecht: Springer Netherlands, pp. 281–295. ISBN: 978-94-015-7372-6. DOI: 10.1007/978-94-015-7372-6_19. URL: https://doi.org/10.1007/978-94-015-7372-6_19.

United Nations Department of Economic and Social Affairs (2020). *United Nations E-government Survey: Digital government in the decade of action for sustainable development*. United Nations.

Utz, Christine, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz (2019). '(Un)Informed Consent: Studying GDPR Consent Notices in the Field.' In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: ACM, pp. 973–990. ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3354212. URL: http://doi.acm.org/10.1145/3319535.3354212.

Van Alsenoy, Brendan (2019). *Data Protection Law in the EU: Roles, Responsibilities and Liability*. Cambridge: Intersentia.

Veltman, Kim H (2001). 'Syntactic and semantic interoperability: new approaches to knowledge and the semantic web.' In: *New Review of Information Networking* 7.1, pp. 159–183.

Verbeek, Peter-Paul (2005). *What things do: Philosophical reflections on technology, agency, and design*. Penn State Press.

Verou, Lea, Amy X. Zhang, and David R. Karger (2016). 'Mavo: Creating Interactive Data-Driven Web Applications by Authoring HTML.' In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST

'16. Tokyo, Japan: ACM, pp. 483–496. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984551. URL: http://doi.acm.org/10.1145/2984511.2984551.

Vila, Tony, Rachel Greenstadt, and David Molnar (2003). 'Why We Can'T Be Bothered to Read Privacy Policies Models of Privacy Economics As a Lemons Market.' In: *Proceedings of the 5th International Conference on Electronic Commerce*. ICEC '03, pp. 403–407.

Welke, Larry (1980). 'The Origins of Software.' In: *Datamation* December.

Wing, Jeannette M. (Mar. 2006). 'Computational Thinking.' In: *Commun. ACM* 49.3, pp. 33–35. ISSN: 0001-0782. DOI: 10.1145/1118178.1118215. URL: http://doi.acm.org/10.1145/1118178.1118215.

Winner, Langdon (1980). 'Do artifacts have politics?' In: *Daedalus*, pp. 121–136.

World Justice Project (2020). *Rule of Law Index*. URL: https://worldjusticeproject.org/sites/default/files/documents/WJP-ROLI-2020-Online_0.pdf.

Yates, JoAnne (1995). 'Application Software for Insurance in the 1960s and Early 1970s.' In: *Business and Economic History*, pp. 123–134.

Yeh, Ron, Chunyuan Liao, Scott Klemmer, François Guimbretière, Brian Lee, Boyko Kakaradov, Jeannie Stamberger, and Andreas Paepcke (2006). 'ButterflyNet: A Mobile Capture and Access System for Field Biology Research.' In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: ACM, pp. 571–580. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124859. URL: http://doi.acm.org/10.1145/1124772.1124859.

Zachmann, Mark S. (1983). 'Context MBA: Half a Step in the Right Direction.' In: *PC Magazine* 2.1.

Zaki, Zulkifly M., Peter M. Dew, Mohammed H. Haji, Lydia M. S. Lau, Andrew Rickard, and Jennifer Young (2011). 'A User-orientated Electronic Laboratory Notebook for Retrieval and Extraction of Provenance Information for EUROCHAMP-2.' In: *2011 IEEE Seventh International Conference on eScience*, pp. 371–378. DOI: 10.1109/eScience.2011.58.

Zaki, Zulkifly M., Peter M. Dew, Lydia M. S. Lau, Andrew R. Rickard, Jenny C. Young, Tahir Farooq, Michael J. Pilling, and Chris J. Martin (2013). 'Architecture design of a user-orientated electronic laboratory notebook: A case study within an atmospheric chemistry community.' In: *Future Generation Computer Systems* 29.8, pp. 2182 –2196. ISSN: 0167-739X. DOI: 10.1016/j.future.2013.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0167739X1300071X.

Zanfir-Fortuna, Gabriela (2020). *Why data protection law is uniquely equipped to let us fight a pandemic with personal data*. URL: https://pdpecho.com/2020/04/06/why-data-protection-law-is-uniquely-equipped-to-let-us-fight-a-pandemic-with-personal-data/ (visited on 07/08/2020).

Zittrain, Jonathan (2009). 'The Generative Internet.' In: *Communications of the ACM* 52.1, 18–20. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1435417.1435426.

Zuiderveen Borgesius, Frederik J, Sanne Kruikemeier, Sophie C Boerman, and Natali Helberger (2017). 'Tracking Walls, Take-It-Or-Leave-It Choices, the GDPR, and the ePrivacy Regulation.' In: *European Data Protection Law Review* 3.3, pp. 353–368. DOI: 10/gfsh4x.

Östberg, P., A. Hellander, B. Drawert, E. Elmroth, S. Holmgren, and L. Petzold (2012). 'Reducing Complexity in Management of eScience Computations.' In: *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 845–852. DOI: 10.1109/CCGrid.2012.72.