The Application and Its Consequences for Non-Standard Knowledge Work

Midas Nouwens
Digital Design & Information Studies
Aarhus University
midasnouwens@cavi.au.dk

Clemens Nylandsted Klokmose
Digital Design & Information Studies
Aarhus University
clemens@cavi.au.dk

ABSTRACT

Application-centric computing dominates human-computer interactions, yet the concept of an application is ambiguous and the impact of its ubiquity underexplored. We unpack "the application" through the lens of non-standard knowledge work: freelance, self-employed, and fixed-term contract workers who create knowledge in collaboration with a wide variety of stakeholders on a per-project basis. Based on interviews with fourteen participants we describe 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. We trace these experiences to common characteristics of applications, such as update processes, interface symmetries, application-document relationships, and operating system and hardware dependencies. By empirically and analytically focusing on "the application", we reveal the implications of the current application-centric computing paradigm and discuss how variations within this model create qualitatively different human-computer interactions.

Author Keywords

Application-centric computing; application paradigm; knowledge work; non-standard work

ACM Classification Keywords

H.5.0. Information Interfaces and Presentation (e.g. HCI): General

INTRODUCTION

Application-centric computing has dominated human-computer interactions for the past forty years. Arguably 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" [17] (e.g., a spreadsheet, a word processor, an email client, a web browser). Traditionally, an application takes the form of an isolated piece of software installed on a computer that operates on certain file types using a collection of predetermined programs – this is as true for Xerox PARC's 1974 Bravo editor as for Microsoft's Word 2016. Since then, mobile, web, and cloud based services have emerged as another type of software referred to as applications (or *apps*). Although application-centric computing has been criticised before (e.g., [33], [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 1970s, when large (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 continued to evolve. The labour landscape in post-industrial economies has shifted from predominately full-time, permanent, open-ended employment (i.e., *standard employment*) [35], to one with increasingly more short-term, flexible, project-based forms of labour (i.e., *non-standard employment*) [10]. 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 [34]. 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.

CHI 2018, April 21–26, 2018, Montréal, QC, Canada.

ACM ISBN 978-1-4503-5620-6/18/04...$15.00
http://dx.doi.org/10.1145/3173574.3173973

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 paper 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.

RELATED WORK

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" [9] and the NLS/Augment system [12] 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 create universally accessible "personal working files" [13] through the framework of Open Hyperdocument Systems and Dynamic Knowledge Repositories [14].

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 [21], architectural work [39], and office management [40]. Other approaches have focused on particular activities within knowledge work, such as file sharing [32], collaborative writing [44], using common data repositories [30], or web activity [41].

However, few have considered "the application" as the unit of analysis. The ones that have (e.g., [19]) are mostly rooted in the standard employment perspective and studied workers in large organisations with access to enterprise level and domain-specific applications.

Non-Standard Work

Historically, studies on technology in work practices were situated in corporate contexts. In third wave HCI [5], 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 [8] and the struggle these workers face with maintaining their mobile offices [42].

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 [31]. 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) [28]. Conform to these developments, HCI literature has engaged with the new types work performed under non-standard contracts – such as the gig economy [2], the platform economy [27], on-demand work [43], and crowd work [20] – but has paid less attention to existing professions (e.g., journalism) under these conditions.

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. [29] 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. [16] 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. [27] 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 [11] 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.

Applications

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

Norman, in The Invisible Computer [33], 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 remained (mostly) editable in-place [22]. Microsoft's *Object Linking and Embedding* and Apple's *OpenDoc* [36] 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 [38], 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 [1] 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"* [24]. 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. [3] 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 [23] 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.

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.

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).

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 [28].

Data Collection

The interviews were semi-structured and the questions open-ended. Inspired by Flanagan's critical incident technique [15], 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.

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.

Analysis

The interviews were analysed using thematic analysis, a qualitative analysis method agnostic to theoretical perspectives [7]. 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 [6]. 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.

RESULTS

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 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.

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.

The Value in Applications

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

Skill-Based Value

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

> *InVision* [1] *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* [2] 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)

[1] A web app for creating clickable, interactive prototypes based on static images.

[2] A web app for designing, prototyping, and documenting interfaces.

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:

> *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.

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.

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.

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.

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.

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.

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.

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).

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.

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.

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.

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.

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.

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 document 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 [4], 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.

Webstrates [25] 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. [25] 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.

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 [37], 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 collaboration 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"* [45]. 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 [26].

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 [11] 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 [18] 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.

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 economic 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.

ACKNOWLEDGEMENTS

This work has been funded by the Aarhus University Research Foundation. We thank the participants for their generosity; Susanne Bødker, Henrik Korsgaard, and Vanessa Thomas for their comments on earlier drafts; and the anonymous reviewers for their valuable feedback and discussion.

REFERENCES

1. 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)*. ACM, New York, NY, USA, 413–422. DOI: http://dx.doi.org/10.1145/319950.323231

2. 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)*. ACM, New York, NY, USA, 4599–4616. DOI: http://dx.doi.org/10.1145/3025453.3025974

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)*. ACM, New York, NY, USA, 211–220. DOI: http://dx.doi.org/10.1145/1124772.1124805

4. 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)*. ACM, New York, NY, USA, 45–53. DOI: http://dx.doi.org/10.1145/120782.120787

5. 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)*. ACM, New York, NY, USA, 1–8. DOI:http://dx.doi.org/10.1145/1182475.1182476

6. Richard E. Boyatzis. 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development*. SAGE Publications, Inc.

7. Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. DOI: http://dx.doi.org/10.1191/1478088706qp063oa

8. Brown Brown and Kenton O'Hara. 2003. Place as a Practical Concern of Mobile Workers. *Environment and Planning A* 35, 9 (2003), 1565–1587. DOI: http://dx.doi.org/10.1068/a34231

9. Vannevar Bush. 1945. As we may think. *Atlantic Monthly* 176 (1945), 101–108.

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

11. 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)*. ACM, New York, NY, USA, 4997–5002. DOI: http://dx.doi.org/10.1145/2858036.2858343

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

13. Douglas C. Engelbart. 1988. Computer-supported Cooperative Work: A Book of Readings. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter Toward High-performance Knowledge Workers (Reprint), 67–78. http://dl.acm.org/citation.cfm?id=49504.49507

14. 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)*. ACM, New York, NY, USA, 143–156. DOI: http://dx.doi.org/10.1145/99332.99351

15. John C Flanagan. 1954. The critical incident technique. *Psychological bulletin* 51, 4 (1954), 327.

16. 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)*. ACM, New York, NY, USA, 1632–1643. DOI: http://dx.doi.org/10.1145/2858036.2858476

17. Bengt Goransson, 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)*. ACM, New York, NY, USA, 133–136. DOI: http://dx.doi.org/10.1145/29933.30872

18. 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)*. ACM, New York, NY, USA, 309–312. DOI: http://dx.doi.org/10.1145/1958824.1958870

19. 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)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 122–129. DOI: http://dx.doi.org/10.1145/223904.223920

20. 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 Computing Systems (CHI '13)*. ACM, New York, NY, USA, 611–620. DOI: http://dx.doi.org/10.1145/2470654.2470742

21. Marina Jirotka, Charlotte P. Lee, and Gary M. Olson. 2013. Supporting Scientific Collaboration: Methods, Tools and Concepts. *Comput. Supported Coop. Work* 22, 4-6 (Aug. 2013), 667–715. DOI: http://dx.doi.org/10.1007/s10606-012-9184-0

22. Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey. 1989. The Xerox Star: A Retrospective. *Computer* 22, 9 (Sept. 1989), 11–26, 28–29. DOI: http://dx.doi.org/10.1109/2.35211

23. Victor Kaptelinin and Liam J. Bannon. 2012. Interaction Design Beyond the Product: Creating Technology-Enhanced Activity Spaces. *Human–Computer Interaction* 27, 3 (2012), 277–309. DOI:http://dx.doi.org/10.1080/07370024.2011.646930

24. David Karger. 2007. Haystack: Per-User Information Environments Based on Semistructured Data. In *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*, Victor Kaptelinin and Mary Czerwinski (Eds.). MIT Press, Cambridge, MA, USA, Chapter 3, 49–100.

25. Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 280–290. DOI: http://dx.doi.org/10.1145/2807442.2807446

26. 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)*. ACM, New York, NY, USA, 71–79. DOI: http://dx.doi.org/10.1145/2940299.2940314

27. 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)*. ACM, New York, NY, USA, 4331–4343. DOI: http://dx.doi.org/10.1145/3025453.3025515

28. Irene Mandl, Maurizio Curtarelli, Sara Riso, Oscar Vargas, and Elias Gerogiannis. 2015. *New Forms of Employment*. Vol. 2. Publications Office of the European Union.

29. David Martin, 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)*. ACM, New York, NY, USA, 224–235. DOI: http://dx.doi.org/10.1145/2531602.2531663

30. 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)*. ACM, New York, NY, USA, 2461–2470. DOI: http://dx.doi.org/10.1145/2556288.2557042

31. Manos Matsaganis, Erhan Özdemir, Terry Ward, and Alkistis Zavakou. 2016. *Non-Standard Employment and Access to Social Security Benefits*. Technical Report. Directorate-General for Employment, Social Affairs and Inclusion, European Commission.

32. 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)*. ACM, New York, NY, USA, 763–766. DOI: http://dx.doi.org/10.1145/1753326.1753438

33. Donald A. Norman. 1998. *The Invisible Computer*. MIT Press, Cambridge, MA, USA.

34. OECD. 2015. *In It Together: Why Less Inequality Benefits All*. OECD Publishing, Paris. DOI: http://dx.doi.org/10.1787/9789264235120-en

35. German Federal Ministry of Labour and Social Affairs. 2015. Green Paper Work 4.0. *Strategic Notes* (2015).

36. Kurt Piersol. 1994. Under the Hood: A Close-Up of OpenDoc. *BYTE* 19 (1994), 183–188. http://archive.li/zPfWW

37. Rob Pike and Brian W Kernighan. 1984. The UNIX System: Program Design in the UNIX Environment. *AT&T Bell Laboratories Technical Journal* 63, 8 (1984), 1595–1605.

38. Jef Raskin. 2000. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional.

39. Kjeld Schmidt and Ina Wagner. 2004. Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning. *Computer Supported Cooperative Work (CSCW)* 13, 5 (01 Dec 2004), 349–408. DOI: http://dx.doi.org/10.1007/s10606-004-5059-3

40. Abigail J. Sellen and Richard H.R. Harper. 2003. *The Myth of the Paperless Office*. MIT Press, Cambridge, MA, USA.

41. 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)*. ACM, New York, NY, USA, 227–234. DOI: http://dx.doi.org/10.1145/503376.503418

42. Norman Makoto Su and Gloria Mark. 2008. Designing for Nomadic Work. In *Proceedings of the 7th ACM Conference on Designing Interactive Systems (DIS '08)*. ACM, New York, NY, USA, 305–314. DOI: http://dx.doi.org/10.1145/1394445.1394478

43. Rannie Teodoro, 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)*. ACM, New York, NY, USA, 236–247. DOI: http://dx.doi.org/10.1145/2531602.2531680

44. Bill Tomlinson, Joel Ross, Paul Andre, Eric Baumer, Donald Patterson, Joseph Corneli, Martin Mahaux, Syavash Nobarany, Marco Lazzari, Birgit Penzenstadler, Andrew Torrance, David Callele, Gary Olson, Six Silberman, Marcus Stünder, Fabio Romancini Palamedi, Albert Ali Salah, Eric Morrill, Xavier Franch, Florian Floyd Mueller, Joseph 'Jofish' Kaye, Rebecca W. Black, Marisa L. Cohn, Patrick C. Shih, Johanna Brewer, Nitesh Goyal, Pirjo Näkki, Jeff Huang, Nilufar Baghaei, and Craig Saper. 2012. Massively Distributed Authorship of Academic Papers. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA, 11–20. DOI: http://dx.doi.org/10.1145/2212776.2212779

45. Langdon Winner. 1980. Do artifacts have politics? *Daedalus* (1980), 121–136.