Exploring cognitive waste and cognitive load in software development - a grounded theory

Daniel Helgesson
Dept. of Computer Science
Lund University
daniel.helgesson@cs.lth.se

Abstract

This paper provides theoretical account of a critical/grounded theory study in industrial software engineering/development settings. We extend our research into the 'critical ethnography' domain, drawing on Rosen, by revisiting previously collected data using social theory as a theoretical filter. We explore the relations and underlying tensions between alienation and *absence of* user centered design in regards to software development tools design using a critical lens. We further experiment with representation of qualitative data and analysis using commentary excerpt units. We present a grounded theory account of the consequences of absent user centred design activities in regards to digital tools in industrial software engineering. We conclude that not deploying *user centered design* when developing software development tools is really (stressing *really*) bad waste management policy for a multitude of reasons.

Research paradigm: design science

Epistemological position: pragmatist

Methodology: case study

Method: grounded theory/(critical) ethnography

Analysis: abductive/iterative

Data collection: interviews, semistructured

Theoretical underpinnings: distributed cognition, cognitive load theory, cognitive load drivers, perspectives, social theory, marxism

1. Intro

As we have previously stated (Helgesson, 2021) the meaning of 'software engineering' is dual. One meaning is literal, i.e. 'the process of engineering software', the other is a scientific engineering discipline aimed providing software development practitioners with scientific evaluations, suggestions and knowledge in regards to software development processes and tools. The actual phrase was coined at the first NATO conference on the matter in the late sixties (Naur & Randell, 1969) . Despite the fact that "personell factors" (Naur & Randell, 1969) was brought up as important factor in software engineering activities at the seminal conference, and that software engineering as such was identified as "socioculturally constituted phenomenon" in the mid nineties (Bertelsen, 1997) it has as a scientific discipline and academic field been dominated by positivism and quantitative methodologies largely focused on non-human factors (Lenberg, Feldt, & Wallgren, 2015) .

Further 'odd' and/or 'novel' research in terms of methodology within the community are often met with a 'so what' response from reviewers (Sharp, Dittrich, & de Souza, 2016) rather than an 'now thats interesting' (Davis, 1971). That being said being said, considerable efforts within the field have been made to draw on qualitative methodology from social sciences, e.g. (Runeson, Höst, Rainer, & Regnell, 2012), (Stol, Ralph, & Fitzgerald, 2016), (Sharp et al., 2016).

The core phenomena we study are *cause*, and subsequent *consequence*, of cognitive load (i.e. 'roughly mental effort' (Helgesson, 2021)) as a consequence of digital work environment and corresponding tools in the software industry. We have previously used grounded theory (Charmaz, 2014) (Bryant, 2017) approach in three studies resulting in a taxonomy of 'cognitive load drivers' (Helgesson, Engström, Runeson, & Bjarnason, 2019), an ethnographic study of cognitive load in a distributed cognitive

setting (Helgesson, Appelquist, & Runeson, 2021), and a synthesis of previous research and extant literature (Helgesson & Runeson, 2021).

Purpose of paper – this paper, an industrial case study, is positioned as a halfway marker in (4th research cycle/paper) in an extended (five year) grounded theory project on cognitive load (Helgesson & Runeson, 2021), as a consequence of digital work environment in software industry. In this specific paper we explore 'critical traditions' especially 'critical ethnography'(Prasad, 2018) in software engineering by revisiting and re-analysing our aggregated dataset in regards to the sensitising concepts (Charmaz, 2014) of 'cognitive sustainability' and 'cognitive productivity' we noted in previous work (Helgesson & Runeson, 2021). Further we deploy concepts from 'social theory' (Luastsen, Larsen, Nielsen, Ravn, & Sörensen, 2017) (i.e. primarily 'alienation') as analytical filter/lens. Additionally, the analysis serves as a vehicle for open-ended exploration on how one can 'write' qualitative research in the software engineering research community using 'commentary excerpt units' (Rennstam & Wästefors, 2018).

2. Background

2.1. Cognitive load as phenomenon

Cognitive load (i.e. 'roughly mental effort' (Helgesson, 2021)) is per definition inherent in all forms of cognitive work. The limits of the human mind in terms of information processing and corresponding bandwidth has been well known for more than fifty years (Miller, 1956). Most, if not all, activities in software development are inherently cognitively loaded (Sedano, Ralph, & Péraire, 2017). We use the term 'cognitive load driver'(Helgesson et al., 2019) to describe the causal nature of cognitive load in regards to digital tools and work environment in the software industry. In the context of this paper we look at cognitive load from a more general cognitive work environment perspective as described by Gulliksen, Lantz, Walldius, Sandblad, and Åborg (2015). Kirsh (2000) describes cognitive overload[1] in work settings.

2.2. Cognitive load in software engineering

Cognitive load in software engineering has, to date, largely been investigated using quantitative methodology (Gonçales, Farias, & da Silva, 2021) (Helgesson, 2021). We let (Müller & Fritz, 2016) and (Fritz & Müller, 2016) serve as contemporary examples. In addition to our qualitative studies (Helgesson et al., 2019), (Helgesson et al., 2021), (Helgesson & Runeson, 2021), Sedano et al. (2017) used grounded theory to investigate different forms of 'waste' in software development, finding 'extraneous cognitive load' being one aspect.

2.3. Marxist/critical traditions and social theory in software engineering

In software engineering marxist traditions (Prasad, 2018) appear to have had very small impact as of yet (Melegati & Wang, 2021) . The authors discuss 'Critical theory' as a 'research paradigm' rather than 'a tradition'. The study mentions Hilderbrand et al. (2020) on 'gender'. We have further found (Vorvoreanu et al., 2019) and (Burnett, Peters, Hill, & Elarief, 2016) on the same issue.

Finally, marxist approaches to explore software related phenomena can be found in other fields of research - e.g. (Fuchs & Sevignani, 2013), (Pfeiffer, 2014), (Nygren & Gidlund, 2016) and (Krüger & Johanssen, 2014). See also: cyberspace Froomkin (2002) on 'cyberspace' and Söderberg (2011) on 'hacking', and (D'Ignazio & Klein, 2020) on 'data science' .

Use of social theory in relation to software engineering has been studied in (Ralph, Chiasson, & Kelley, 2016) and (Lorey, Ralph, & Felderer, 2022) respectively.

[1] See also Simon (1971): "In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it."

3. Method

3.1. Grounded theory and epistemological position

In this paper, an industrial case study, we use grounded theory[2] largely as prescribed by Charmaz (2014) and Bryant (2017)[3][4].

We deploy an ethnographic approach (Charmaz & Mitchell, 2001) using distributed cognition (Hollan, Hutchins, & Kirsh, 2000) and concepts from 'social theory' (Luastsen et al., 2017) (primarily 'alienation') as theoretical observational filters. We venture further into the 'critical' ethnography tradition (Prasad, 2018) using extant literature [5].

The work reported in this paper is conducted within the research paradigm of 'design science' (Runeson, Engström, & Storey, 2020). We see grounded theory as a general purpose qualitative method for generating 'theory' (Abend, 2008) in order to describe a problem in practice, and we take a pragmatist (Rorty, 1979) epistemological grounded theory position (Bryant, 2017) [6].

3.2. Research goal and research questions

Given the exploratory nature of grounded theory it is common for studies informed by grounded theory to start of with "an open ended research goal" (Stol et al., 2016) or "initial research questions that evolve throughout the study" (Charmaz, 2014).

Our initial **research goal** was:

– To explore previously recorded data in regards to cognitive load in software engineering using 'grounded theory ethnography' (Charmaz & Mitchell, 2001) from a 'critical ethnography' (Prasad, 2018) angle using the sensitising concepts of 'cognitive sustainability', 'cognitive productivity' and 'cognitive sustainability' we noted in previous work in conjunction with 'social theory' (Luastsen et al.,

[2]Grounded theory is an exploratory (Stol et al., 2016) qualitative method, by some heralded as one of the (if not 'the') most important qualitative methods to appear in the scientific toolbox to date (Bryant, 2017). The aim of grounded theory is to provide a framework for generating 'theory' (Abend, 2008) from (largely) qualitative data (Charmaz, 2014). Core elements in most grounded theory method sections written by phd students is a lengthy (and somewhat pretentious) segment on the history of grounded theory (Bryant, 2017). Having written three of those ((Helgesson et al., 2021), (Helgesson & Runeson, 2021), (Helgesson, 2021)) we will not waste further space on this matter in this context. We see grounded theory as a general purpose iterative qualitative method/vehicle for generating theory from largely qualitative data abductively (or inductively). The guidelines provided by Charmaz (2014) helps us keep control of data and ensures methodological rigour (Gioia, Corley, & Hamilton, 2013) and transparency. No more. No less.

[3]In order to be very precise here: Bryant (2017) does not consider his methodological approach as a singular version of grounded theory in contradiction to that of Charmaz (2014), but rather as a complement. The main difference lies in epistemological position. In order to rule out the relativisation that a constructivist position allows for, and may result in, "'ultimate caricature of postmodernism", Bryant advocates for a 'pragmatist' (Rorty) epistemological position. So from a methodological perspective we rely on Charmaz, while framing our epistemological within the epistemological framework of Bryant since we want to close the door on relativisation.

[4]It should also be noted that Bryant (2017) highlights that the tension does not lie in a (false) dichotomy between 'qualitative' and 'quantitative' methodologies (and analysis and data). This is not really that hard to comprehend. How would it be possible to make sense of quantitative findings without qualitative analysis? Instead Bryant highlights that the main epistemological tension lies between 'objectivist' and 'constructivist' stand points. They further offers a solution to this tension by taking the middle ground of 'pragmatism', thereby taking a position immune to constructivist epistemological critique of objectivism as the core tenet in 'pragmatism' is that it, itself, is inherently 'fallible' and 'contingent'. With a background in 'sciences of the natural'(Simon, 1969) constructivism (and corresponding risks of relativisation) is is not an easy epistemological position to uphold and We thus lean toward pragmatism.

[5]Reading Glaser from a distance (i.e. not by using close reading, but reading explicitly): "...reading and use of literature is not forsaken in the beginning of a grounded theory project. It is vital to be reading and studying from the outset of the research, *but in unrelated fields*" (Glaser, 1992, p. 35). This is a clear indication that the use of literature and abduction has been relevant in traditional grounded theory for almost 30 years. From our perspective the role of abduction has been rhetorically underplayed and the matter of literature absence has equally been rhetorically overplayed in grounded theory discourse. See also (Martin, 2019) and Dey (1999), (1993).

[6]Given the many variants and interpretations on grounded theory (Bryant, 2019) guidelines on grounded theory often highlight the need to be very specific on describing what version of grounded theory and what epistemological position is being used in order to avoid 'method slurring' and avoiding criticism for deploying a "rhetorical sleight of hand", e.g. (Stol et al., 2016). We will, however, not venture further into this discussion at this point.

2017) in order to see what a 'critical' 'perspective' (Helgesson & Runeson, 2021) of cognitive load in software engineering might consist of.

This evolved into the following **research questions**:

1 – What problems (psychosocial and cognitive) can be observed, in regard to a software development tool (and its replacement) from a user perspective by means of social theory and distributed cognition?

2 – What are the underlying root causes of these problems, from a 'critical' angle?

3 – Can we further our understanding of 'cognitive load drivers' and 'cognitive waste' by means of deploying a 'critical' analysis?

3.3. Study design consideration

In this study we revisit and use qualitative data previously collected, and reason abductively (Martin, 2019)[7] in regards to these observations using extant literature from social sciences. So, the design is labelled as 'flexible explorative case study' (Runeson et al., 2012) using 'grounded theory'(Charmaz, 2014). We take deploy a 'critical ethnography' (Prasad, 2018) angle in this study, drawing on (Rosen, 2000).

In closing, in regards to the labelling of this research – we investigate tool use, and consequences thereof, within a culture in an exploratory fashion, so the research approach is, in our humble opinion the inherently ethnographic. With that said, we are perfectly fine with leaning on the Chicago school tradition root of grounded theory (Charmaz, 2014), (Bryant, 2017) and simply label the study 'an industrial case study, using grounded theory and an ethnographic approach/data set'.

3.4. Case description

We are operating within 'case study'-methodology (Runeson et al., 2012). The case in this paper is a large, 1000+ developers, international multi-site software development organisation. The object under study is cognitive load, as a consequence of digital work environment in software industry, as experienced by software developers. Over all we study the object from an individual as well as a distributed perspective. In this specific study the unit of analys is the individual engineer but in the distributed cognitive context (Hollan et al., 2000).

3.5. Data collection & data set construction

In this study we rely on 3 semi structured interviews (1 test engineer, 1 development engineer, 1 tools responsible). Interviews were recorded, and transcribed by first author who also translated them into English. Interviews were conducted in another language.

The first interview was conducted in the first field trip. We then approached the 'gatekeeper'/company contact point and were provided access to a person responsible for tool activities within the company in order to provide some background. This lead to an informal discussion lasting for about an hour during which notes were taken. It was not recorded on account on the person being uncomfortable in recording a session that they was unprepared for.. This was later recorded in a formal interview. The engineer was approached and asked if willing to volunteer for an interview to add a different perspective on the previous discussions for 'triangulation' purposes (van Maanen, 1979). We also added extant literature from 'social sciences' (Luastsen et al., 2017) (e.g. 'alienation') to the data set in order to explain the phenomena we encountered after the open coding.

The interviews were fractured at the transcriptions, using emacs text line editor and .csv formatted spatially separating the interviewers and the interview subject. Time codes were added at key passages.

3.6. Analysis

Coding was executed in two stages ('open' and 'focused') as suggested by Charmaz (2014). 'Open coding' was executed, in multi-pass fashion to allow for 'analytical bracketing' (Rennstam & Wäste-

[7] See also (Bryant & Charmaz, 2019), (Bryant, 2017). Further, the definition offered by Alvesson and Sköldberg (2018, p. 4-8) is very succinct.

fors, 2018) [8], using the transcripts printed on paper and Pilot-V disposable reservoir pens in red and blue. Codes for relevant 'incidents'(Glaser, 1978) were written on the printed transcripts. Coding was executed 'chunk-by-chunk' rather than actual 'line-by-line'.

The codes were then transferred to Post-It stickers (set A) as prescribed by Bryant (2017) and grouped, thus forming 'categories'. The 'categories' were then named. Prior to 'open coding' we had used (Prasad, 2018)) in order to sensitise ourselves with the 'critical' tradition and to achieve some level of "theoretical sensitivity"(Glaser, 1978) for the phenomena under study. We then produced a few 'memos' (Charmaz, 2014) outlining the findings of the 'open coding' stage. Prior to this stage we read (D'Ignazio & Klein, 2020) in order to achieve some level of senzitation in regards to contemporary critical discourse in software settings. We then read, and coded using Post-It-stickers, 'Social theory' by Luastsen et al. (2017) to further explore our findings (Set B).

'Focused coding' was done in three stages: firstly we grouped the two sets (A and B) of Post-It stickers and based on this we produced another 'memo' into which fractured interview transcripts were added forming lengthy 'excerpt commentary units' (Rennstam & Wästefors, 2018). This process was also done in multi-pass fashion to allow for 'analytical bracketing'(Rennstam & Wästefors, 2018), but the purpose was to generate what van Maanen (1979) refers to as 'second order themes' (i.e. not only allowing the 'what' and 'how' but also the underlying ethnographically important 'why' (Sharp et al., 2016)) to emerge. Following this process we started writing the analysis using 'memoing'. Finally, these 'memos' were fused into one that largely makes up the analysis section of this paper.

The rendering of our theory (Charmaz, 2014) was done in the same fashion as the one we generated in previous work (Helgesson et al., 2021), using Post-It stickers and an A1 cardboard sheet. We have previously found this technique extremely useful as it allows for a level of tactile feedback and interaction impossible if one uses software. The theory was then transferred into digital format[9]. It has been constructed from the actual 'memos' and presented in a seminar.

3.7. Literature review

We did informal/preliminary literature surveys prior and post open coding. Following the focused coding and theory generation we extended this using a similar strategy as in (Helgesson & Runeson, 2021). We queried ACM from 2010 to 2013 with queries in regards to 'marxism/critical theory' in conjunction with 'alienation' [10] as well as 'social theory' [11]. In doing so we found 48 and 36 papers respectively. Having done a reading of the abstracts we found no papers directly relevant to this study. But, we conclude that the findings of Melegati and Wang (2021) are correct - the impact of the critical tradition in the area of software development/engineering has been small. We also similarly queried IEEE [12] [13] with similar queries finding 99 and 87 titles respectively. From an abstract reading we could find no text directly relevant for this study.

[8] See also: (Gearing, 2004), (Tufford & Newman, 2012) for in-depth discussions in regards 'bracketing'.

[9] While some reviewers have previously lamented on the absence of artefacts in regards to 'coding', 'memoing' and 'theory generation' we are in agreement of Bryant (2017) – these artefacts are private. The paper is the actual outcome, and artefact, of the study. With that said, we are, of course happy to allow for audit of our anonymised transcripts and field notes – please contact first author for details

[10] [[All: "marxism"] OR [All: "critical theory"] OR [[All: "marxism"] AND [All: "alienation"]]] AND [[All: "software development"] OR [All: "software engineering"]] AND [E-Publication Date: (01/01/2010 TO 12/31/2023)]

[11] [All: "social theory"] AND [[All: "software development"] OR [All: "software engineering"]] AND [E-Publication Date: (01/01/2010 TO 12/31/2023)]

[12] (("Full Text Metadata":"marxism" OR "Full Text Metadata":"critical theory") OR ("Full Text Metadata":"marxism" AND "Full Text Metadata":"alienation")) AND ("Full Text Metadata":"Software Development" OR "Full Text Metadata":"Software Engineering")

[13] ("Full Text Metadata":"social theory") AND ("Full Text Metadata":"Software Development" OR "Full Text Metadata":"Software Engineering")

4. Analysis

4.1. Theory and theorising

We draw on excerpt commentary units (Rennstam & Wästefors, 2018) here; 'Tell', 'Show', 'Explain' but intentionally convoluted and meta-level. The two engineers describe similar phenomena and the tool responsible provides the underlying background. 'Categories' are contained in [hard brackets], explanations in <sharp brackets>, narrative is denoted with '–talking dash' and dialogue is contained within "quotes".[14] is work in progress. The elements we explore in this theory are shown in Figures 1.

We see our 'theory' (Abend, 2008) as an abstract account of events that have taken place within a software development organisation that allows for 'theorisation' (Sutton & Staw, 1995),(Weick, 1995) in regards to consequences of poor tooling within the software development industry.

4.2. Absent UCD, cognitive waste, cause and consequence for the individual digital worker

The first facet of the theory we generate from the data set explores absence of user centred design in relation to digital tools and cognitive waste from the perspective of the individual digital worker. The 'categories' we choose to explore are: **absent ucd**, **missing functionality**, **cognitive waste**, **frustration**, **coping strategies** and **alienation**.

The first facet of the theory, the individual perspective, is visualised in Figure 1, and each 'category' is explored and further explained in a corresponding subsection below.

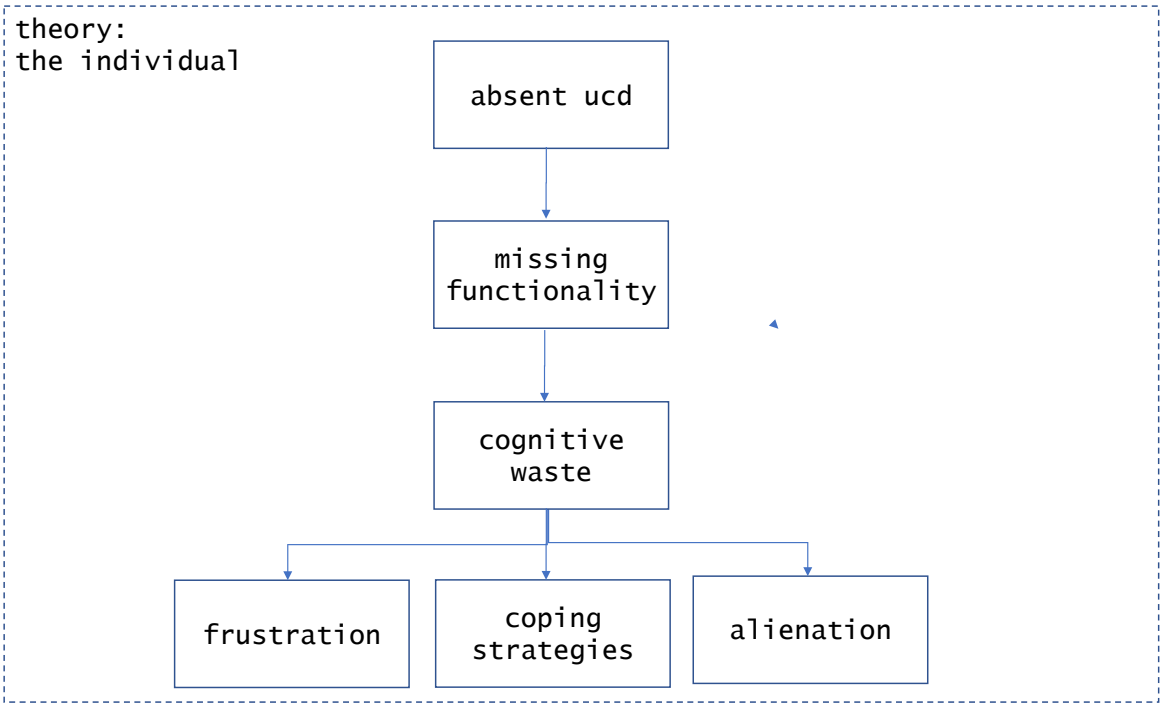

Figure 1 – Theory – consequence for the individual worker caused by absent UCD

4.2.1. cognitive waste

4.2.2. frustration

The following discussion between the interviewer and the informant, reveals that the developer is acutely aware of the problems associated with the tools with which they are provided by the corporation.

"So... what digital tools do you use?"

[14] We use 'categories'(Charmaz, 2014) rather than 'codes' or 'themes' as we have aggregated the 'codes' into 'categories' (of 'codes') and we will be visualising 'categories' in our substantive theory.

“Well – ’Tool 1’, ’Tool 2’ and.... (<trying to remember>)... well, actually I try to avoid all tools if possible!’ They are a pain in the butt to work with. I can’t come up with any good things to say about them[**alienation**]! Starting off with ’Tool 1’: If you are to file an issue, I have to fill in a lot of input fields. About twenty of them[**waste**]. For every field you have to think for one minute[**cog waste**], then you can figure out how *fun* <sarcasm>[**frustration**] writing issues is....”

We clearly see that the effort that the developer need to spend on filing ’issues’ is clearly identified as ’waste’ and that it causes frustration.

4.2.3. absent ucd

4.2.4. alienation

The following two comments from developers display a connection between the absence of user centered design activities and alienation of employees.

”I have a feeling that the software developers haven’t been asked what requirements they have[**no ucd**]. Instead the requirements come from, I don’t really know but I assume, project people and operator managers etc. And they live in a different world than we do[**alienation**].”

”At this company there is an affinity for adding stuff. But not for removing them...This company is extremely poor at that. If *they*[**alienation**] were to start *counting how much time we spend* on this[**waste**] they would probably get shocked.´´

The way the developers use ’*they*’ clearly indicates that individuals see themselves as distanced, and estranged from their employer.

4.2.5. missing functionality

This segment described the missing functionality in the tool discussed at detail. It focuses on two aspects, the plethora of unnecessary information and selections that needs to be supplied when reporting an ’issue’ (a defect report) and missing search functionality preventing the users to navigate the system efficiently. It has been shorted considerably to fit page limitation (currently it only discusses one issue, the interaction, not the missing search functionality).

The following passage in which the interviewer discusses the system with one of the engineers identifies the first source of waste, that the user has no support for recalling his/her most common selections. Instead they have to tediously selected from very long scroll lists in the user interface.

“So there is no *support* [**missing functionality**] that allows you to *see just your* common selections?”

“No, and that would be really good to have[**no support**]. ”

The need for the missing functionality is obvious from the perspective of the developer. Again we see the alienation and the frustration the poor tooling causes. “With one new field?” “No. Three new fields <sarcasm>. Why couldn’t they have removed the other fields[**no support**]? (<implying that redundant work is necessary>)”

“But you have to fill in every field? Or you can’t [**oppression**] file your issue?”

“No, exactly... <sarcasm>[**frustration**] – there is no point in looking at the ’old stuff’ (<the fields that are no longer used>), but you still need to fill them in. At this company there is an affinity for adding stuff. But not for removing them... and ’clean up’. This company is extremely poor at that. If *they*[**alienation**] were to start *counting how much time we spend* on this[**waste**] they would probably get shocked. But it is not as easy as simply filing an issue in ’Tool 1’, once you have done that you must find it again, and go into it and ’do modify’ as you must add what projects you want the issue tagged for. This can’t be done *while you actually file* the issue. So you *can’t even file all information is relevant in one go*. So you file, save, find it *again*[**waste**], select ’modify’, enter the additional info and save it *again*[**waste**]. “That sounds harsh.” “I get *paid*[**alienation**] for this. It is not as bad as it could be. But it could be handled a lot better [**frustration**]...”

Switching over to the other engineer we see that he/she is well aware of the same issues discussed in the previous segment.

"... one thing that strike me in the earlier interviews... All these fields in 'Tool 1' that need to be entered, all variants that exist..."

"A lot of different tabs with a lot of different fields. And I am expected to fill out a subset of these, but it is hard to understand which fields that should be filled out. And what information is expected to be entered. So it appears as there has been requirements coming from a lot of different directions... We need this, and that, and that... And while I'm sure that a lot it is relevant for the projects, it is not relevant for me <as a developer> **[alienation]**"

"So you don't really know what <information> you are expected to hand over <enter into 'Tool 1'>?

"Exactly. Sometimes it is just ridiculous **[frustration]** - a lot of these fields are designed under the premise that we are working with an error, but 'Tool 1' is used for development as well, and to fill out 'in what version of the system the error was detected' is really not applicable but still you have to fill out these fields."

The developer here clearly indicates that the user does not know neither what information to supply, neither where nor why. This largely confirms what was discussed in the previous section.

"Something else that was indicated was that some of these fields were made obsolete/replaced by new fields, yet they remained."

"Correct. Another thing that has occurred is that something new <kind of data> is invented and a field is reused for this purpose. There was 'Found in' and 'Found during' and in 'Found during' you were supposed to fill out what team you were part of. Or something along those lines. Simply because a field was reused. These kind of problems are quite common - you don't really know how to use the system, what information to enter."

"That it is counter intuitive?"

"Counter intuitive, yes. It gives very little guidance of what you are supposed to do. And it is very easy to do the wrong thing. To enter a state that you don't understand how to exit <refers to state of issue not state of program>)."

The consequences of absent user design is clearly visible to the developer. The tool, and corresponding tasks, have become 'counter intuitive' and makes the user error prone,

4.2.6. coping strategies

"... and to search for information in 'Tool 1' is a problem?"

"Yes... I have solved in this way[**coping strategy**]... I get a mail for every issue in 'Tool 1' that is created for my team, and I save these. And I search in these mails because searching in 'Tool 1' for something that is handled and closed is not something I have ever been able understand how to do. And this is actually something we struggle with in newer systems as well. That you get an error report and you think 'didn't we have an issue like that six months ago?' - then you would like to find out what caused it, and how it was solved. But once an issue has been closed... it does exist somewhere in the database... but it is really hard to locate, it is really difficult to search for.. Searching for everything that I have been involved in for instance, or... It might be possible to do if you are really good at creating queries, but it is quite hard for normal users <lit. ordinary mortals>. "

4.3. Absent UCD, cognitive waste, cause and consequence for the cause and consequence for the corporation

The analysis originally also contained a theoretical account of the corporate perspective of this specific case/tool, based on an interview with the person responsible for development and maintenance of the system, and it successor. The person fully acknowledged the issues discussed by the users, and provided

an explanation on why the situation had been allowed to deteriorate. For space reasons we only present a brief overview.

The root cause was absent organisational ownership, compounded by a confounding factor – namely company culture and cultural differences, as well as a managerial lack of understanding of user needs. As a consequence the tool was very inefficient, on account of missing functionality. In addition to the corresponding cognitive and temporal/fiscal waste stemming from the absent UCD, the person also noted that the data quality and reliability (of the data that could be extracted from the system) was very low.

While having observed the aforementioned problems, and the fact that the tool had been voted 'the worst tool within the organisation' by one of the development organisations the rationale of the business case used to replace the existing tool was simply licensing fees rather than lacklustre functionality and reliability. As a consequence the same mistakes were made in the customisation of the replacement – ultimately resulting in a similarly inefficient tool. Again...

4.4. Conclusion

There is a complete absence of understanding of what the user needs from the tool, since no user centered design steps are taken to ensure that it functions properly and serves the needs of the organisation and the users. Only the management perspective of the tool is taken into consideration when designing and implementing it. This is the *complete opposite* of core tenets of human factors engineering practice (Gulliksen et al., 2015) (Wickens, Hollands, Banbury, & Parasuraman, 2015).

In fact it can be noted that Wickens et al. (2015, p. 1) explicitly state that human factors engineering came about "...just after World War II when experimental psychologists were called in to help understand *why pilots were crashing perfectly good aircraft* (Fitts & Jones, 1947)..." It is well known within the company that 'Tool 1' is extremely unpopular among the users in software production, yet it is only changed for short term profitability reasons (lower licensing fees). The frustration it causes, the cognitive and temporal waste it causes is not considered in the business case, neither is the poor quality of data – that renders it useless as an analytical tool – which is one of the main reasons it is there in the first place.

5. Discussions in regard to RQ's

1 – What problems (psychosocial and cognitive) can be observed, in regard to a software development tool (and its replacement) from a user perspective by means of social theory, distributed cognition and "perspectives' respectively? – We focus on psychosocial rather than cognitive issues in this study. We see several aspects of such issues in connection to what we have read in social theory (e.g. 'alienation' and 'frustration',). These, to some extent correspond to what Gulliksen et al. (2015) describes. The "psychological distress" as described by Sedano et al. (2017) is clearly visible in our data set as well.

2 – What are the underlying root causes of these problems, from a 'critical' and cognitive perspectives, respectively? – The underlying root causes of these problems, from a 'critical' perspective can mainly be attributed to absence of user centred design when developing digital tools as a consequence of absent mandate/responsibility and organisational issues.. From a cognitive perspectives, it is mainly related to the presence of 'poorly functioning digital tools' (Håkansson & Bjarnason, 2020) in conjunction with the limits of the human mind (Wickens et al., 2015) (Miller, 1956). The underlying root cause is absence of user centered design (Wickens et al., 2015)[15].

3 – Can we further our understanding of 'cognitive sustainability', 'cognitive productivity' and 'cognitive waste' by means of deploying a 'critical perspective'? – Yes. Indeed we see that we further our understanding 'cognitive productivity' and 'cognitive waste' by means of a 'critical' perspective.

6. Validity/Generalisability

GT studies are commonly evaluated based on the following criteria (Charmaz, 2014) (Stol et al., 2016):

[15]The phenomena of poorly designed (from user centered design/interaction design perspective) software development tools is well known within the software engineering community – it is referred to as 'developers in their own dog-food'

Credibility: *Is there enough data to merit claims of the study?* – Yes. While the data set the theory is grounded in is limited, it still is indicative of a relevant problem.

Originality: *Does the results offer new insight?* – Hopefully, yes. It provides an inside account on the consequences of absence of User Centered Design in the design process of software development tools. It further shows that the there is no tension in relation between cognitive sustainability and efficiency respectively; rather that the tension lies between cognitive sustainability/efficiency and corporate lack of understanding of the need of user centred design.

Usefulness: *Is the theory generated relevant for practitioners?* – As of yet, we do not know. With that said, it is a novel way of explaining the intangible cognitive waste, drain and frustration that occur as a consequence of poorly functioning tools. So hopefully yes.

Resonance: *Does the theory resonate with practitioners?* – from the limited feedback received at the time of writing, yes.

Ethnography is tricky, van Maanen (1979) lists several ways in which the researcher can be ¨¨misled" by informants [16] . In this case we don't see that as an issue. The data set triangulates our findings well What we witnessed largely agreed with what the informants let us in on. We thus consider ourselves "explicitly and extensively informed" (Rosen, 2000).

In regards to general criticism of single case generalisation we humbly point to Anzai and Simon (1979): "It may be objected that a general psychological theory cannot be supported by a single case. One swallow does not make a summer, but one swallow does prove the existence of swallows. And careful dissection of even one swallow may provide a great deal of reliable information about swallow anatomy.".

In the end of the introduction Rosen (2000) cites the dissertation of Kunda (unpublished), to make the point that: "...ethnography is the only human activity in the social sciences. As a method it is not divorce from the modes of experience that I consider human, that is, it is not divorced from my 'reality'.It is therefore one of the few ways of doing research that speaks the 'truth' as I understand it.". We couldn't agree more.

7. Emerging concepts for future studies

In addition to the generated theory, we made some additional observations that we will shortly present and theorise around.

7.1. learning/unlearning

When discussing systems in general with one user we noted that there might be something interesting at play when systems are changed/replaced by something smilar, yet different. The person stated that when a tool was replaced it became very difficult to adapt the mental model of how to operate the system.

Allowing for some theorisation here it is not farfetched to think that when a user has to make an active choice/recall on 'how' to perform an activity (depending on what tool is being used) this will result in unnecessary cognitive load, since what was previously an instinctive recall now has become an active choice. This would not be far fram what Rasmussen (1983) observed, that response to system stimuli is dependent on whether it is a 'skill', 'rule' or 'knowledge'.

7.2. cognitive work & labour

When discussing cognitive load it is interesting to note that it is, similarly to 'load' in physics, essentially momentary. In order to fully understand cognitive waste, it would make sense to reason in terms of cognitive work as a temporal aggregation of cognitive load (similarly to the relation between 'power'/'load' in physics and 'energy'/'work'). Similarly the process of exerting cognitive load for wage could/should be denoted 'cognitive labour'.

[16] See also: (Briggs, 1986) and (Agar, 2008)

8. Discussion

The problem with poorly functioning software development tools is rather well known within the industry , to the point that the term 'developers eat their own doogfood' was been coined decades ago. In this case, however, the issue is rather that 'engineers eat manager dogfood', in a sense. The issues with the tool in question are well known within the organisation, yet there is seemingly little interest in doing something about it. Here we might draw similarities with early industrialisation, where workers were provided tools by industrialists and User Centered Design and ergonomics lay a long time into the future.

When discussed in course seminars the 'mundanity' of the story were reflected on by students from varying disciplines. Given the high level of identification in the kafka-esque story displayed by audience at seminars, this might suggest a wider applicability – and a wider problem[17].

In light of the post conference publication of PPIG we will pend the discussion section post conference presentation.

9. Acknowledgements

We wish to thank librarian Andeas Karman for (once and again) helping out with database queries, and esteemed supervisor prof. Per Runeson for input and proof read (as well as putting up with the grounded theory shenanigans).

10. References

Abend, G. (2008). The Meaning of 'Theory'. *Sociological Theory*, *26*(2), 173–199.

Agar, M. H. (2008). *the Professional Stranger An informal Introduction to Ethnography* (2nd ed.). Bingley, UK: Emerald Group Publishing Ltd.

Alvesson, M., & Sköldberg, K. (2018). *Reflexive Methodology - New Vistas for Qualitative Research* (3rd ed.). London, UK: SAGE Publications.

Anzai, Y., & Simon, H. A. (1979). The Theory of Learning by Doing. *Psychological Review*, *86*(2), 124–140.

Bertelsen, O. W. (1997, November). Toward A Unified Field Of SE Research And Practice. *IEEE Software*, *14*(6), 87–88.

Briggs, C. L. (1986). *Learning how to ask - A sociolinguistic appraisal of the role of the interview in social science research*. Cambridge ; New York: Cambridge University Press.

Bryant, A. (2017). *Grounded Theory and Grounded Theorizing – Pragmatisism in Research Practice*. Oxford, UK: Oxford University Press.

Bryant, A. (2019). *The varieties of grounded theory* (1st ed.). London, UK: SAGE Publications Ltd.

Bryant, A., & Charmaz, K. (2019). *The SAGE Handbook of Current Developments in Grounded Theory*. London, UK: SAGE Publications.

Burnett, M., Peters, A., Hill, C., & Elarief, N. (2016, May). Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 2586–2598). San Jose, California, USA: Association for Computing Machinery. doi: 10.1145/2858036.2858274

Charmaz, K. (2014). *Constructing Grounded Theory* (2nd ed.). London, UK: SAGE Publications.

Charmaz, K., & Mitchell, R. (2001). Grounded Theory in Ethnography. In *Handbook of Ethnography.* London, UK: SAGE Publications.

Davis, M. S. (1971, June). That's Interesting!: Towards a Phenomenology of Sociology and a Sociology of Phenomenology. *Philosophy of the Social Sciences*, *1*(2), 309–344.

Dey, I. (1993). *Qualitative Data Analysis: A user-friendly guide for social scientists*. London, UK: Routledge.

[17] At the time of writing (May 2023) there has been a discussion in media e.g. https://www.expressen.se/ledare/patrik-kronqvist/politikerna-skryter-men--sverige-ar-ett-u-land/ in regards to poorly functioning systems in the wake of the ongoing digitalisation of society which also supports that, albeit not applicable to software development tools and organisations, this is becoming a societal issue at a large scale.

Dey, I. (1999). *Grounding Grounded Theory - Guidelines for Qualitative Inqury*. San Diego, California, USA: Academic Press.

D'Ignazio, C., & Klein, L. F. (2020). *Data Feminism* (1st ed.). Cambridge, Mass. USA: MIT Press.

Fritz, T., & Müller, S. C. (2016, March). Leveraging Biometric Data to Boost Software Developer Productivity. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (Vol. 5, pp. 66–77).

Froomkin, A. M. (2002). Habermas@Discourse.Net: Toward a Critical Theory of Cyberspace. *Harvard Law Review*, *116*(3), 749–873.

Fuchs, C., & Sevignani, S. (2013, June). What Is Digital Labour? What Is Digital Work? What's their Difference? And Why Do These Questions Matter for Understanding Social Media? *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, *11*(2), 237–293. doi: 10.31269/triplec.v11i2.461

Gearing, R. E. (2004, December). Bracketing in Research: A Typology. *Qualitative Health Research*, *14*(10), 1429–1452. (Publisher: SAGE Publications Inc) doi: 10.1177/1049732304270394

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013, January). Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. *Organizational Research Methods*, *16*(1), 15–31. Retrieved from `https://doi.org/10.1177/1094428112452151` (Publisher: SAGE Publications Inc) doi: 10.1177/1094428112452151

Glaser, B. G. (1978). *Theoretical Sensitivity*. CA, USA: Sociology Press.

Glaser, B. G. (1992). *Emergence vs Forcing - Basics of Grounded Theory Analysis*. CA, USA: Sociology Press.

Gonçales, L. J., Farias, K., & da Silva, B. C. (2021, August). Measuring the cognitive load of software developers: An extended Systematic Mapping Study. *Information and Software Technology*, *136*, 106563. doi: 10.1016/j.infsof.2021.106563

Gulliksen, J., Lantz, A., Walldius, Å., Sandblad, B., & Åborg, C. (2015). *Digital arbetsmiljö, en kartläggning (RAP 2015:17)* (Tech. Rep.). Retrieved from `https://www.av.se/arbetsmiljoarbete-och-inspektioner/kunskapssammanstallningar/digital-arbetsmiljo-kunskapssammanstallning/`

Helgesson, D. (2021). *Exploring grounded theory perspectives of cognitive load in software engineering* (Unpublished doctoral dissertation).

Helgesson, D., Appelquist, D., & Runeson, P. (2021). A grounded theory of cognitive load drivers in novice agile software development teams. In *Unpublished manuscript.* Retrieved from `http://arxiv.org/abs/2107.04254`

Helgesson, D., Engström, E., Runeson, P., & Bjarnason, E. (2019). Cognitive Load Drivers in Large Scale Software Development. In *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 91–94). Piscataway, NJ, USA: IEEE Press. doi: 10.1109/CHASE.2019.00030

Helgesson, D., & Runeson, P. (2021). Toward grounded Theory perspectives in Software Engineering. In *Ppig'21.*

Hilderbrand, C., Perdriau, C., Letaw, L., Emard, J., Steine-Hanson, Z., Burnett, M., & Sarma, A. (2020, June). Engineering gender-inclusivity into software: ten teams' tales from the trenches. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 433–444). Seoul, South Korea: Association for Computing Machinery. doi: 10.1145/3377811.3380371

Hollan, J., Hutchins, E., & Kirsh, D. (2000, June). Distributed Cognition: Toward a New Foundation for Human-computer Interaction Research. *ACM Trans. Comput.-Hum. Interact.*, *7*(2), 174–196.

Håkansson, E., & Bjarnason, E. (2020, August). Including Human Factors and Ergonomics in Requirements Engineering for Digital Work Environments. In *2020 IEEE First International Workshop on Requirements Engineering for Well-Being, Aging, and Health (REWBAH).* doi: 10.1109/REWBAH51211.2020.00013

Kirsh, D. (2000). A Few Thoughts on Cognitive Overload. *Intellectia*(30), 19–51.

Krüger, S., & Johanssen, J. (2014, September). Alienation and Digital Labour—A Depth-Hermeneutic Inquiry into Online Commodification and the Unconscious. *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, *12*(2), 632–647. doi: 10.31269/triplec.v12i2.548

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015, September). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*, *107*, 15–37.

Lorey, T., Ralph, P., & Felderer, M. (2022). Social science theories in software engineering research. In *Proceedings of the 44th international conference on software engineering* (p. 1994–2005). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3510003.3510076` doi: 10.1145/3510003.3510076

Luastsen, C. B., Larsen, L. T., Nielsen, M. W., Ravn, T., & Sörensen, M. P. (2017). *Social Theory - A Textbook* (1st ed.). New York, NY, USA: Routledge.

Martin, V. (2019). Using Popular and Academic Literature as Data for Formal Grounded Theory. In *The SAGE Handbook of Current Developments in Grounded Theory.* London, UK: SAGE Publications.

Melegati, J., & Wang, X. (2021, May). Surfacing Paradigms underneath Research on Human and Social Aspects of Software Engineering. In (pp. 41–50). IEEE Computer Society. doi: 10.1109/CHASE52884.2021.00013

Miller, G. A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*, *63*(2), 81–97.

Müller, S. C., & Fritz, T. (2016, May). Using (Bio)Metrics to Predict Code Quality Online. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (pp. 452–463). (ISSN: 1558-1225) doi: 10.1145/2884781.2884803

Naur, P., & Randell, B. (1969, January). *Software engineering: Report on a conference sponsored by the nato science committee* (Tech. Rep.). Scientific Affairs Division, NATO.

Nygren, K. G., & Gidlund, K. L. (2016). *The Pastoral Power of Technology. Rethinking Alienation in Digital Culture*. Brill. (Pages: 396-412 Section: Marx in the Age of Digital Capitalism) doi: 10.1163/9789004291393_013

Pfeiffer, S. (2014, September). Digital Labour and the Use-value of Human Work. On the Importance of Labouring Capacity for understanding Digital Capitalism. *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, *12*(2), 599–619. doi: 10.31269/triplec.v12i2.545

Prasad, P. (2018). *Crafting Qualitative Research: Beyond Positivist Traditions*. Routledge.

Ralph, P., Chiasson, M., & Kelley, H. (2016). Social theory for software engineering research. In *Proceedings of the 20th international conference on evaluation and assessment in software engineering.* New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2915970.2915998` doi: 10.1145/2915970.2915998

Rasmussen, J. (1983, May). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(3), 257–266. doi: 10.1109/TSMC.1983.6313160

Rennstam, J., & Wästefors, D. (2018). *Analyze! Crafting your Data in Qualitative Research* (1st ed.). Lund, Sweden: Studentlitteratur.

Rorty, R. (1979). *Philosophy and the mirror of nature* (1st ed.). Prinston, NJ, USA: Princeton University Press.

Rosen, M. (2000). *Turning Words Spinning Worlds* (Vol. 25). Harwod.

Runeson, P., Engström, E., & Storey, M.-A. (2020). The Design Science Paradigm as a Frame for Empirical Software Engineering. In M. Felderer & G. H. Travassos (Eds.), *Contemporary Empirical Methods in Software Engineering* (pp. 127–147). Cham: Springer International Publishing. doi: 10.1007/978-3-030-32489-6_5

Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons.

Sedano, T., Ralph, P., & Péraire, C. (2017, May). Software Development Waste. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)* (pp. 130–140). doi: 10.1109/ICSE.2017.20

Sharp, H., Dittrich, Y., & de Souza, C. R. B. (2016, August). The Role of Ethnographic Studies in Empirical Software Engineering. *IEEE Transactions on Software Engineering*, *42*(8), 786–804.

Simon, H. A. (1969). *The sciences of the artificial* (2nd ed.). Cambridge, USA: MIT Press.

Simon, H. A. (1971). Designing organizations for an information-rich world. In M. Greenberger (Ed.), *Computers, communication, and the public interest* (pp. 40–41). Baltimore. MD: The Johns Hopkins Press.

Stol, K.-J., Ralph, P., & Fitzgerald, B. (2016, May). Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (pp. 120–131). doi: 10.1145/2884781.2884833

Sutton, R. I., & Staw, B. M. (1995). What Theory is Not. *Administrative Science Quarterly*, *40*(3), 371–384. (Publisher: [Sage Publications, Inc., Johnson Graduate School of Management, Cornell University]) doi: 10.2307/2393788

Söderberg, J. (2011). *Free software to open hardware: critical theory on the frontiers of hacking* (No. 17). Gothenburg: University of Gothenburg, Dept. of Sociology.

Tufford, L., & Newman, P. (2012, January). Bracketing in Qualitative Research. *Qualitative Social Work*, *11*(1), 80–96. (Publisher: SAGE Publications) doi: 10.1177/1473325010368316

van Maanen, J. (1979). The Fact of Fiction in Organizational Ethnography. , 13.

Vorvoreanu, M., Zhang, L., Huang, Y.-H., Hilderbrand, C., Steine-Hanson, Z., & Burnett, M. (2019, May). From Gender Biases to Gender-Inclusive Design: An Empirical Investigation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–14). New York, NY, USA: Association for Computing Machinery.

Weick, K. E. (1995). What Theory is Not, Theorizing Is. *Administrative Science Quarterly*, *40*(3), 385–390. (Publisher: [Sage Publications, Inc., Johnson Graduate School of Management, Cornell University]) doi: 10.2307/2393789

Wickens, C. D., Hollands, J. G., Banbury, S., & Parasuraman, R. (2015). *Engineering Psychology & Human Performance*. Psychology Press.