Cognitive Load Drivers in Large Scale Software Development

Helgesson, Daniel; Engström, Emelie; Runeson, Per; Bjarnason, Elizabeth

Published in:
2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)

DOI:
10.1109/CHASE.2019.00030

2019

Document Version:
Early version, also known as pre-print

Cognitive Load Drivers in Large Scale Software Development - Preprint

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

Emelie Engström
Dept. of Computer Science
Lund University
Lund, Sweden
emelie.engstrom@cs.lth.se

Per Runeson
Dept. of Computer Science
Lund University
Lund, Sweden
per.runeson@cs.lth.se

Elizabeth Bjarnason
Dept. of Computer Science
Lund University
Lund, Sweden
elizabeth.bjarnason@cs.lth.se

***Abstract*—Software engineers handle a lot of information in their daily work. We explore how software engineers interact with information management systems/tools, and to what extent these systems expose users to increased cognitive load. We reviewed the literature of cognitive aspects, relevant for software engineering, and performed an exploratory case study on how software engineers perceive information systems. Data was collected through five semistructured interviews. We present empirical evidence of the presence of cognitive load drivers, as a consequence of tool use in large scale software engineering.**

***Index Terms*—Cognition, Cognitive Load, Software Development, Software Engineering, Software Development Tools, Software Engineering Tools, Industrial Case Study**

I. Introduction

Software engineering is a socio-technical endeavour where the technical side of the phenomena seems to be more studied than the social side [1], and as a consequence knowledge of a cognitive/ergonomic perspective of software development, and the tools associated with these activities, appears rather small. Further, we see no clear indications of a significant impression on the software engineering community in terms of understanding the cognitive work environment of software engineers [2] [3].

In a 2002 dissertation, Walenstein observed that there is a need for cognitive reasoning in the design process of software development tools, and further that there has been little research done in the area [2], a claim largely substantiated by Lenberg et al. [1].

More recently, in a 2015 report 'Digital Work Environment', Gulliksen et al. made an effort to analyse the societal consequences of large-scale digitalisation of human labour, in general [4]. In the report the authors present a literature survey, providing updated insight into the research area. The survey found only 36 relevant articles. In addition, the authors also present a taxonomy of 'Cognitive work environment problems'.

In this study we aim to explore, and establish, a broader understanding of the cognitive work environment of software engineers and the cognitive dimensions of the tools used. Specifically, we aim to explore *cognitive load*, induced on users by information systems or tools. We present results from an exploratory industrial case study based on thematic analysis of interviews, as well as a literature overview. Our contribution lies in presenting in vivo observation of cognitive problems associated with tool use in large-scale software engineering.

II. Research questions

The purpose of this study is to gain insight into the problem domain of cognitive load, primarily as a consequence of tool use, in large scale software development. Hence it is exploratory in nature, and focuses on two tools central for communication and knowledge management at the case company.

The overall exploratory purpose is refined into two research questions:

RQ1 Which types of cognitive load drivers can be observed in large-scale software engineering, primarily as a consequence of tool use?

RQ2 How do software engineers perceive the identified cognitive load drivers in their digital work environment?

The research questions are anchored in software engineering and cognitive science literature, and addressed by interviewing practitioners. The first question uses the cognitive literature as a lens, while presenting empirical observations from the interview material. The second question reports the interviewees' perception of problems found in RQ1.

III. Method

We conducted a four stage case study, using a flexible design [5]; consisting of *literature review, interview study, extended interview study*, and *knowledge synthesis*. To mature the knowledge, we iterated reviewing literature, conducting interviews, transcribing and analysing data. Figure 1 describes the study.

The case company is an international corporation, the studied division develops consumer products in the Android ecosystem. The software is embedded in handheld devices. The studied development site of the company has 1000 employees and developers work in cross-functional teams using an agile development process. The development environment is primarily based on the toolchain associated with the Android ecosystem.

The case study is informed by five interviews. We started with three interviews of people from a test team – a test manager (i1), and two testers (i2, i3). After an initial analysis

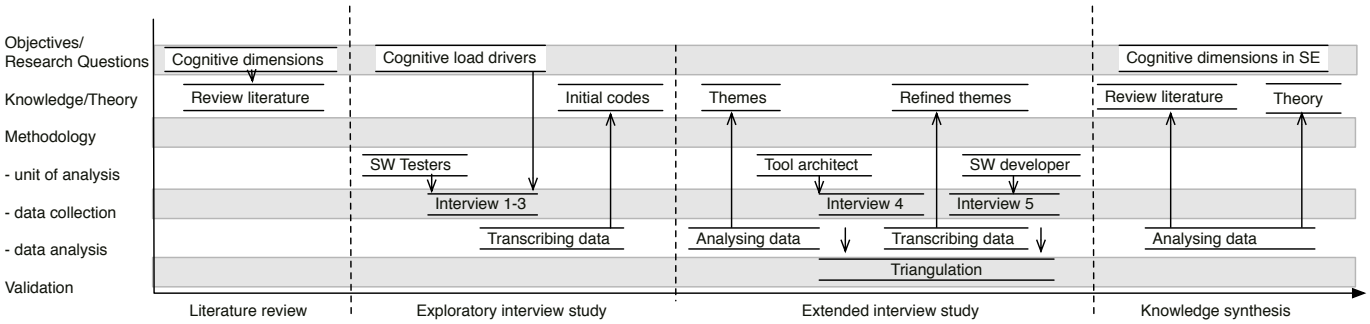

Fig. 1. A description of how the case study evolved in terms of objectives, knowledge and research activities

of the of interviews, one additional interview (a tool architect, i4) was conducted to to provide background. A final interview (a software developer, i5) was added to validate the findings. All interviews were recorded, transcribed by the first author, then iteratively coded and analysed by the authors collectively.

The first three interviews were semistructured, following an interview guide[1]. These three interviews were conducted by the first and fourth authors collectively. The fourth interview was specifically centered around the two most discussed tools of interviews 1-3. The main architect of these tools was selected for this interview, due to being able to provide background on the development history, as well as rationales on certain design decisions. This interview was conducted by the first author. The fifth person interviewed was selected as he had worked with the defect management system since its introduction ten years ago. This interview followed the protocol of the first three. The interview was conducted by the first author.

The data was analysed as outlined in Figure 1, using thematic analysis. A set of initial codes were identified during the transcription of the first three interviews by the first author. These codes were then used to create a set of themes in two iterations. The fourth and fifth interviews were used to validate and extend the initial set of interviews. The themes were then refined, and reapplied to all five interviews to extract information on cognitive load drivers. The main iterations of the thematic analysis were executed by three authors collectively.

The classification of cognitive load drivers was then validated against the classification presented by Gulliksen et al.

IV. Literature overview and analysis

A. Purpose and strategy

The purpose of our literature overview is twofold as we aim to: i) provide an overview of cognitive research, relevant and related to software engineering; and ii) present qualitative observations from relevant cognitive literature, not specifically targeting software engineering, that can serve as a step stone for further research. It should be noted that we, by no means, claim that this is a formal, nor systematic, literature study [6]. What we present are, essentially, *qualitative* findings from an *exploratory* literature study, executed as a part of an *exploratory* case study.

[1]see appendix for details

B. Exploratory survey of cognitive research related to Software Engineering

In her keynote at ASE 2018 [7], Murphy presented an updated example of cross industrial/academic software engineering research bordering on cognition, which emphasizes the relevance of context in software engineering. Arguably, the most researched cognitive aspect of software engineering, historically, is program comprehension – a comprehensive overview of past, current and future research directions of program comprehension is presented by Siegmund [3], while Sharafi et al. systematically reviewed software engineering research, that use eye-tracking [8]. Human aspects of software engineering was studied by Lenberg et al. [1], claiming that less than 5% of the articles studied "a 'soft' or human-related topic".

While there is a lot of research on Human Computer Interaction in general, very little is specifically looking at software development and tools. However, there are examples of articles on usability of software tools, e.g., Myers et al. [9], Dillon and Thomas [10]; as well as on design of software tools, e.g., Holzinger [11].

In conclusion, the problem with cognitive research in Software Engineering appear as being twofold i) not very much research has been done beyond code comprehension, and ii) the research has rarely been executed in the real world, 'in vivo', context.

C. Practical implications of human cognitive limitations

It has, since Miller's 'The Magical Number Seven, Plus or Minus Two' [12] from the 1950's, been generally accepted that the human working memory is finite. Accepting that 'the capacity of the human working memory' and cognitive bandwidth (i.e. the amount of cognitive load a human mind can process) are closely related, it can be deduced that unnecessary cognitive, or mental, load is likely to decrease the cognitive bandwidth, which over time translates into 'throughput' or 'work'. Miller's findings are generally accepted, although later research shows the actual bandwidth being lower than Miller proposed [13]. Herein lies the main rationale for studying cognitive load drivers - if all tools (or tasks) induce cognitive load on the subject, keeping the undesired cognitive load to a minimum will allow for more of the cognitive bandwidth being used for relevant chores and not practical waste.

V. Results

We analysed the interview data, partially in the light of the general cognition literature summarised above, and identified different types of load drivers (RQ1), and perceived problems in the digital work environment (RQ2).

A. Identified cognitive load drivers (RQ1)

During the analysis of the interview material, we identified factors mentioned by interviewees, which were interpreted as having an impact on the cognitive load of the developers, i.e. being *cognitive load drivers*. Then, in the coding, these factors were grouped into *items*, which characterise the cognitive load, e.g. lack of adaptation of a tool to the work task. The items were then grouped into *themes* and finally clustered into the object of the cognitive load drivers[2].

The two main clusters of load drivers in the thematic analysis are: *Tool* – cognitive load directly associated to use of tools; and *Information* – cognitive load associated to information management, information flow and information load respectively. In addition we found a third cluster: *Work process* – cognitive load derived from processes and organisational aspects of the workplace.

1) Tool: The aspects of cognitive load associated to *tools*, are of three kinds, namely *intrinsic, delay* related, and *interaction*.

Intrinsic aspects include lack of functionality, manifesting itself as use of tools with poor *adaptability/suitability* to the purpose for which they were intended, including lack of *functionality*. For example, the search functionality is not up to date, which makes it hard to find old defect reports. Further, *lack of stability or reliability* prevent users from trusting the tools. '...if it crashes all data is lost' (i2). *Overlap* and *lack of integration* prevent users from working efficiently, as it causes redundant work (e.g. copying information from one tool/system to another). Finally, *comprehension*, i.e. understanding what is going on when using the tool, is a cognitive load factor – 'It is not obvious how it should be used.' (i3).

Delay is an absence of response in a *tool* or system. It can be observed in our interviews as a state of forced concentration when the user is forcing information to remain in working memory while waiting for the tool to respond; and it can be observed in terms of system downtime. We observed cognitive load from *response* delays at the micro level, as noted by interviewee i3: 'But when it is non responsive you loose focus. You can't just stay focused.' Further, *downtime* on the macro level scale beyond a few seconds effectively prevents all work with the system. 'It has happened that [the tool] has been down for longer periods, especially after upgrades' (i2).

Three different aspects of cognitive loads were identified which considered consequences of *interaction* issues with *tools*. *Unintuitive* implies that the users find the tools hard to use, causing frustration and unnecessary cognitive load to the user as he/she must repeatedly find out how to complete a task. 'I felt that I had to transform my [mental] model to some kind of database model, in order to understand how the tool worked' (i5). *Inconsistent* systems or different parts of a system work rather differently in terms of interaction, causing frustration to the user as he/she repeatedly must determine how to solve a task, in the current context, specifically. *Cumbersome* interaction is when functionality is missing or implemented in such a way that the user is forced to waste energy doing what is considered unnecessary work.

2) Information: The quality of the *information* in itself is the second cluster of cognitive load drivers. The different aspects of cognitive load associated to the *integrity* of information are related to *incompleteness* of information, which causes the user to spend effort in asserting that information is complete. The lack of *reliability* of the information is causing the user to spend effort in asserting that information is correct and up-to-date (e.g. caused by lack of version control). Interviewee i2 stated: 'The main issue is that there is no revision handling/version control'. Finally, the *temporal traceability* of information over time is needed to help a user to bridge a temporal gap in order to assess the current situation, e.g. see if an issue has been reported before, or if an error has been fixed in an earlier release.

The different aspects of cognitive load associated with the *organisation* of *information* is related to *location*, i.e. having a hard time finding information, *retrieval*, i.e. having a hard time retrieving information, *distribution*, i.e. not knowing where to store information or whom to distribute it to, *overview/zoom*, i.e. absence of overview or zoom views cause cognitive load when browsing information. 'There is no overview' (i2). Finally, *structure* is when the information is structured or organised in a cumbersome fashion, e.g. as mentioned by i3, when test results are not saved per test run, something that later cause problems with information accessibility and affects the users' ability to find correct information. The same was indicated by i1, who stated that: 'If I rerun this test project I only see the last result', indicating that the visualisation delta between the current run and the previous run would be beneficial.

3) Work process: The aspects of cognitive load associated to *work process* mainly gravitate around lack of support, manifesting itself in wasted effort; either by doing unnecessary work, or by having to spend effort in finding out how to solve a task in a specific tool or in a certain team. Cognitive load drivers in this cluster are related to *lack of automation*, *wasted effort* (be it unnecessary or redundant), illustrated by one interviewee (i2), stating on mandatory information fields to be filled out even when not needed. Further, *ad hoc* implementation of tools or processes, lead to wasted or inconsistent work process, since they are implemented differently in different parts of the organisation. Finally, *lack of understanding* of the intrinsics of a large organisation can be a load driver itself.

4) Validation of empirical findings: We validated our collection of cognitive load drivers against the set of 'Cognitive work environment problems'[3], identified by Gulliksen et

[2] see appendix for a tabular representation of the result

[3] see appendix for a tabular representation of the validation

al. [4]. In light of that they studied digitalization of work in general, it is quite natural that the cognitive issues identified differ somewhat compared to our findings. That being said, there does not appear to be any contradictions between the two sets, and while not identical, they are largely similar.

B. Perceived problems (RQ2)

The main issue with the test management tool, was missing revision control and absence of revision history. Furthermore, all interviewees had noted that there was no strict ownership, meaning that any user could change test cases and test scripts as they saw fit. The main issue with the defect management tool, apart from cumbersome interaction, was that omission of search functionality - which makes it very hard to find error reports (e.g. duplicated, closed or obsolete error reports). To exemplify, one user used the notification e-mails supplied by the defect management tool to create his own temporal model/history of the error reports handled by his team (i5). Furthermore, as a consequence of the interaction issues with the tool, the quality of the data extracted from the system was perceived to be quite unreliable (i4). The same interview revealed an interesting observation regarding the selection/acquisition of systems. Despite the discontent among the users, and the apparent flaws of the tool, these aspects were not considered part of the business case when the defect management system was replaced. Instead, the sole rationale for the change was, according to the interviewee, that the licensing cost of the new tool/system was significantly lower than that of the old tool.

C. Summary

We conclude that cognitive load drivers are indeed present and have considerable impact in large-scale software engineering, in this specific case, and that the load drivers identified gravitate around three clusters; 'Tools', 'Information' and 'Work process'.

VI. Limitations

The literature review is not complete, in terms of covering all cognitive science literature relevant for software engineering, nor all software engineering literature related to cognition. The goal of the literature review and analysis is to create a foundation for the exploratory case study. Thus it is sufficient to have relevant cognitive science literature as a basis, which we ensure by having some core scholars represented, such as Miller, [12] and Siegmund [3]. In regards of software engineering literature, we observe that there are some aspects of cognition studied, which confirms it being a relevant perspective.

The case study is conducted at one company, with a limited sample size of interviews, limiting its external validity. However, the aim of the study is to explore the field. We have, in all likelihood, missed some cognitive load drivers, that are not present in the case study context. Our validation against Gulliksen et al's taxonomy [4] shows a sound mapping of the cognitive load drivers in a general working context to those found in this study. The authors have long working and collaboration history with the case company, which helps improving the construct validity although it may introduce bias. However, as we only provide evidence of the presence of cognitive load drivers, that does not limit the validity of the findings.

VII. Conclusion

We conclude, that with the exception of research on program comprehension, little is published on cognitive aspects of tool use in software engineering. We also observe that there is indeed work in the cognitive science area relevant for software engineering, which could be used to lay out the theoretical basis for studying and improving software engineering from a cognition point of view. Grounded in the literature in the field, we conclude that we in this exploratory study have found, and presented, empirical evidence of the presence of cognitive load can be observed in large-scale software engineering, RQ1. We further conclude that it is indeed is a problem for practitioners, RQ2. Our classification of the load drivers found gravitates, or clusters, around 'Information', 'Tools' and 'Work/Process'.

This research was financed by projects EASE and ELLIT respectively.

References

[1] P. Lenberg, R. Feldt, and L. G. Wallgren, "Behavioral software engineering: A definition and systematic literature review," *Journal of Systems and Software*, vol. 107, pp. 15–37, Sep. 2015.

[2] A. Walenstein, *Cognitive Support in Software Engineering Tools: A Distributed Cognition Framework*, 2002.

[3] J. Siegmund, "Program Comprehension: Past, Present, and Future," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5, Mar. 2016, pp. 13–20.

[4] J. Gulliksen, A. Lantz, Å. Walldius, B. Sandblad, and C. Åborg, "Digital arbetsmiljö, en kartläggning (RAP 2015:17)," Tech. Rep., 2015.

[5] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case study research in software engineering: guidelines and examples*. Wiley, 2012.

[6] Y. Levy and T. J. Ellis, "A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research," *Informing Science: The International Journal of an Emerging Transdiscipline*, vol. 9, pp. 181–212, 2006.

[7] G. C. Murphy, "The Need for Context in Software Engineering (IEEE CS Harlan Mills Award Keynote)," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 5–5.

[8] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc, "A systematic literature review on the usage of eye-tracking in software engineering," *Information and Software Technology*, vol. 67, pp. 79–107, Nov. 2015.

[9] B. A. Myers, A. J. Ko, T. D. LaToza, and Y. Yoon, "Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools," *Computer*, vol. 49, no. 7, pp. 44–52, Jul. 2016.

[10] B. Dillon and R. Thompson, "Software development and tool usability," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, May 2016, pp. 1–4.

[11] A. Holzinger, "Usability Engineering Methods for Software Developers," *Commun. ACM*, vol. 48, no. 1, pp. 71–74, Jan. 2005.

[12] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information." *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.

[13] N. Cowan, "The Magical Mystery Four: How Is Working Memory Capacity Limited, and Why?" *Current Directions in Psychological Science*, vol. 19, no. 1, pp. 51–57, Feb. 2010.

TABLE I
COGNITIVE LOAD DRIVERS IN SOFTWARE ENGINEERING, STRUCTURED ACCORDING TO THE THEMATIC ANALYSIS.

Main Clusters	Themes	Items	Description
Work process	—	Lack of automation	Absence of automated tool support (e.g. automated testing) forcing the user to do work manually.
		Wasted effort	Unnecessary or redundant work mandated by absence of tool support or by process.
		Ad hoc	Tool support (or processes) implemented differently in different parts of the organisation.
		Lack of understanding	Missing information or support on account of the shifting nature of a large organization.
Tool	Intrinsic	Adaptation/Suitability	Use of tools that are not really suited for the purpose.
		Lack of functionality	Use of tools missing functionality needed to solve a task efficiently.
		Stability/Reliability	Use of tools that suffer from stability or reliability issues.
		Overlap	Use of several tools that can do the same thing, or almost the same thing, in parallel.
		Lack of Integration	Use of several tools, in parallel, that are not (or poorly) integrated, forcing the user to do redundant and/or manual work.
		Comprehension	Actually understanding what needs to be done in the tool in order to complete a task.
	Delay	Response (micro)	Delays in response forcing the user to stay overly focused, putting a strain on short term memory.
		Downtime (macro)	Tools or systems that are completely unresponsive for a longer period than a few seconds/minutes.
	Interaction	Unintuitive	Functionality (or interaction) is implemented in an unintuitive way.
		Inconsistent	Functionality (or interaction) is inconsistently implemented in two different tools or in two different views of the same tool.
		Cumbersome	Functionality (or interaction) is implemented in a way that users find clumsy.
Information	Integrity	Incompleteness	Lack of complete information is causing the user to spend effort in asserting that information is complete.
		Reliability	Lack of reliable information is causing the user to spend effort in asserting that information is correct and up-to-date.
		Temporal traceability	The user needs to bridge a temporal gap in order to assess the current situation.
	Organisation	Location	Where to find the information.
		Distribution	Where to store and whom to distribute the information to.
		Retrieval	How to access the information and retrieve it.
		Overview/zoom	How to navigate the information.
		Structure	How the information is organised.

TABLE II
MAPPING OF OUR MAIN CLUSTERS FROM TABLE I AND GULLIKSEN ET AL'S COGNITIVE WORK ENVIRONMENT PROBLEMS [4]

Cognitive work environment problems	Work process	Tools	Information
Unneccessary cognitive load and interruption of thought process	–	x	–
Unneccessary strain on working memory	–	x	–
Problems orientatiing and lack of overview	–	x	x
Identifying and interpreting information	–	x	x
Decision making/support	–	x	x
Difficulties with time coordination of data	–	x	x
Work processes determined by tools	–	x	–
Many unintegrated information systems	x	x	x
Poor support for learning	x	–	–
Lack of understanding automation	x	–	–
Difficulties with different system modes	N.A.	N.A.	N.A.

Interview Guide: Overview of the Cognitive Support for Testing

1. Introduction: Research topic, Study context, Confidentiality, Audio recording
2. Your background: current role, experience & education (years), past work places

The artefact-related information testers use

- *Info characteristics: type, format*
- *Activities & CRUD of info – who, when, why?*

3. Describe your main work tasks. What information/sources do you use for these?

Tools and cognitive bottle necks

- *Aggregated data views, e.g. in metrics/KPIs, visualisations*
- *Duplicated info*
- *Lack of tool integration (memorising)*
- *Manual analysis required*

4. Which tools & systems do you use for your daily work? For which tasks?
    - Who creates / uses the information in the tools that you use?
    - How do the tools affect your daily work; positive and negative (duplication, memorising, manual analysis)?
    - Do you use any aggregated data views, e.g. visualisation, KPIs? Describe.
    - How can you affect the tools and systems used for your work?

5. Are there systems that you don't work with directly but that you know affect your day-to-day work (e.g. for organising or monitoring your work)? In what way?
    - Who uses these systems? For what?
    - Can you affect them?

6. Over time, what tools & systems have been introduced, replaced or removed?
    - What drives these changes, and how do you feel about them?

Work Environment

7. How do the tools affect your relationship to your colleagues, in the team, in the wider organisation? For example, regarding communication, work load.

8. How do you handle work – life balance (now and in the past)? Is it affected by the tools & systems available?
    1. At work, how do you relate to social media, private e-mails, phone calls etc?
    2. Are you expected to be available outside of work hours? How do you feel about this?
    3. Are there any policies? Monitoring?

Improvements

9. If it was up to you, what kind of feature/tool support would you introduce to better support you in your work?