Neurophysiological Impact of Software Design Processes on Software Developers

Randall K. Minas[1(✉)], Rick Kazman[1], and Ewan Tempero[2]

[1] Shidler College of Business, University of Hawaii at Manoa, 2404 Maile Way, Honolulu, HI 96822, USA
{rminas, kazman}@hawaii.edu
[2] Department of Computer Science, The University of Auckland, 303/38 Princes St, Auckland 1010, New Zealand
e.tempero@auckland.ac.nz

Abstract. Software development often leads to failed implementations resulting from several factors related to individual reactions to software design. Some design metrics give software developers guidelines and heuristics for use in software design. Furthermore, many metrics have been created to measure outcomes in terms of "code quality." However, these guidelines and metrics have been shown only to have a weak relationship and are poorly implemented. This study takes a new approach using tools from cognitive neuroscience to examine the cognitive load and arousal level placed on software engineers while working with different software designs. Specifically, we use electroencephalography (EEG) and skin conductance (SCR) to examine cognitive and emotional reactions to software structure. We propose to examine whether modular design affects levels of cognitive load and arousal. Our findings open the door for future research that combines software engineering and cognitive neuroscience. The potential implications of this study extend beyond optimal ways to structure software to leading the software engineering field to study individual cognition and arousal as a central component in successful software development. This opens a wide array of potential studies in the software engineering field.

Keywords: Cognitive load and performance · Emotion · Electroencephalography · Augmented cognition · Arousal · Software engineering

1 Introduction

Developing software systems is difficult and often leads to failed implementations, resulting from non-completion, reduced confidence, or budgetary overruns [1, 2]. Central to software development is design and development of the code that drives the system. Software engineering is focused on the development of software using a systematic, structured, repeatable approach. These systematic approaches to software development leave an engineer with many decisions on how to organize their software. Researchers in the software engineering field have explored the consequences of different design decisions, such as the choice of identifiers [3], code branching operations [4], and modular structure [5]. The problem, however, is that the evidence of the

D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2017, Part II, LNAI 10285, pp. 56–64, 2017.
DOI: 10.1007/978-3-319-58625-0_4

effectiveness of many of these guidelines and heuristics is, at best, weak. Furthermore, there is plenty of evidence that software developers routinely ignore this advice, even when they are aware of it and know the claimed benefits [6–8]. Much research has focused on the outputs of the development process, or aspects of a software engineer's decision making process. But little research has focused on the affective and cognitive responses software engineers have to the structure—the design—of their software. Taken together, the rate of software development failures and our relative immaturity at dealing with their root causes call for a new approach to understanding the benefits and problems associated with the design of software.

Many disciplines, from marketing to information systems, have begun to apply the tools of cognitive neuroscience to open the "black-box" of the brain, examining areas such as cognitive load, arousal, decision-making, and others [9, 10]. Cognitive neuroscience uses methods such as electroencephalography (EEG), function magnetic resonance imaging (fMRI), and psychophysiological responses (e.g., skin conductance and facial electromyography) to understand how individuals process information, stress levels, cognitive load, and emotion. Some cognitive neuroscience studies have also examined software engineering problems, such as [11] where the author used fMRI to understand program comprehension. However, this study did not elucidate how different types of designs may affect a programmer's cognitive load and arousal. The use of tools from cognitive neuroscience can generate new insights into how software engineers develop software from shedding light on optimal software structure to uncovering the best practices, from a cognitive standpoint, that will lead to better system development and fewer software implementation failures.

The purpose of this study is two-fold. First, it seeks to determine the differences in cognitive load and arousal related to different software designs, identifying from a cognitive and arousal standpoint, which designs lead to lower cognitive load and arousal for the software engineer. Second, and perhaps more importantly, this study seeks to establish a framework from which future software engineering studies can employ the use of cognitive neuroscience methods to gain further insight into the effects individual cognition and emotion have on software engineering efficiency and effectiveness. Our findings should have implications for future software engineering studies and could open a door to a new research agenda within software engineering.

2 Theoretical Background

2.1 Software Engineering and Development Metrics

Much research exists in software engineering that examines different metrics to understand code and design quality. This has been an active research area in software engineering for over four decades (see, for example [12–14]). There are a plethora of metrics that have been proposed to measure the quality of code in Object Oriented (OO) programming, such as CK Metrics [15], LK Metrics [16], and MOOD Metrics [17]. All of these metrics take a different approach to measuring code quality. For example, the MOOD Metrics contains six metrics that examine data encapsulation (e.g., the ability of a program to hide implementation through separate compilation of

modules), information hiding (e.g., visibility of methods and/or attributes to other code), inheritance (e.g., when an object or class is based on another object), and coupling (e.g., the degree of interdependence between modules) [17, 18].

The aforementioned metrics examine the output artifact—the code that has been written—and do not examine the arousal or cognitive load it takes for a software engineer to understand, debug, or update this code. Since there are many ways to structure coding from full object-orientation, to naïve object-orientation, to a relatively unstructured "God Class" [19], the metrics are only providing a relative comparison of the output artifacts. As they give no insight into the cognitive load or arousal level placed on the software engineer, the usefulness of these metrics is limited.

2.2 Cognitive Load and Arousal and Software Engineering

Cognitive load theory is a major theory in the cognitive science literature that provides a framework for investigating cognitive processes, simultaneously considering the structure of information and cognitive architecture that allows learners to process information efficiently [20]. Cognitive load refers to the amount of cognitive resources that are needed to process a given task. Low cognitive load tasks require fewer cognitive resources, while high cognitive load tasks require higher cognitive resources [20]. One of the key cognitive resources that underlies cognition is working memory [21]. It encapsulates both what many consider "short-term memory" and attention. Therefore, working memory is pivotal for both information processing and specific tasks (i.e., software engineering), responsible for encoding information from the environment and retrieving information from long-term memory in order to make sense of it [21–23]. A useful computer analogy for understanding working memory is that it represents the brain's RAM, storing of information currently undergoing processing but limited in its capacity [24].

Working memory is most heavily associated with the frontal areas of cortex, namely Dorsolateral Prefrontal Cortex (DLPFC) [25]. However, it is important to note that neural networks distribute working memory to multiple areas of cortex. Changes in activity in the DLPFC indicate changes in working memory load and attention [26]. In EEG, attenuation of the alpha rhythm over DLPFC indicates increases in working memory load [27]. Cognitive load in working memory (in the frontal and temporal regions of the brain) is reduced when irrelevant information is presented, as less information needs to be maintained for subsequent recall [28]. Skin conductance, a measure of autonomic nervous system arousal, is also closely tied to cognitive load [29]. As cognitive load increases, autonomic system arousal increases [30].

In software engineering, there have been some studies on cognitive load. One study applied design techniques from other disciplines that have found to be low in cognitive load and applied them to software engineering, finding that current visual implementations for software engineers need improvement to lower cognitive load on software engineers [31]. Another study has applied cognitive models to the novice software analyst, finding that novices often have trouble in scoping the problem and poor formation of the conceptual model of the problem domain [32]. However, despite these studies and others, no studies have instituted the tools of cognitive neuroscience to

directly measure cognitive load and arousal during software updating tasks. Our study examines the link between cognitive load, arousal and structured and unstructured software engineering tasks. In cases where the code is highly structured, we believe the individual's cognitive load and arousal will be lower. Moreover, in cases where the code is unstructured, we believe there will be a greater cognitive load and arousal. Therefore, we hypothesize:

H1: An individual's cognitive load and arousal will be higher when there is less structure in the code (e.g., God Class) than in a fully object-oriented design.

3 Method

Our goal in this study is to understand how individual software engineers respond cognitively and emotionally to different software structures. Therefore, the unit of analysis will be the individual. The study participant will be given a software update task to perform in Java using the Eclipse IDE environment.

3.1 Participants

Participants will be undergraduate students from a state university who will receive $75 compensation for their time. The participants were drawn from upper-level computer science courses and had experience with both Java and the Eclipse environment. We aim to collect data on 50 subjects in a repeated measures design.

3.2 Task

We will use an implementation of the game Kalah in the Eclipse environment. Participants will be asked to make a changes to the code to implement different functionality for the game. One task will ask the participant to change the number of initial seeds in the houses from two to three. This is a trivial change. The second task will ask the subject to institute an AI player for the game. This is a more involved change in the code. We will examine cortical changes and skin conductance during each task. The participants will also have practice task to gain familiarity with the game in the Eclipse environment. We use a repeated measures design in this study, so each participant will be exposed to both tasks.

3.3 Treatments

There will be two treatments. One treatment will be the God Class for a program of the game Kalah—a single class that implements all of the game's functions. The second treatment will be a fully object-oriented modular design of the game Kalah. This will be a repeated measures design with participants being exposed to both treatments.

Independent Variable

The independent variable in this study is the task the participant is making changes in the code. There are two levels of difficulty in the task, trivial and non-trivial. We will examine these along with the two treatment levels: God Class and full object-orientation.

Dependent Variables

Our dependent variables are cortical alpha wave activity, autonomic arousal, and emotional valence. These are operationalized using neurological and psychophysiological measures. EEG measures will be collected using a 14-channel headset (Emotiv Systems, San Francisco, CA, USA) with electrodes dispersed over the scalp along the 10–20 system [33] (see Fig. 1). The electrodes will make connection with the scalp surface via felt pads saturated in saline solution. The reference electrodes will be located at P3 and P4 over the inferior, posterior parietal lobule [33]. All other channels will be measured in relation to the electrical activity present at these locations, sampled at 128 Hz. Impedances will be verified and data collected using Emotiv TestBench Software Version 1.5.0.3, which will export it into comma-delimited format for subsequent analysis.

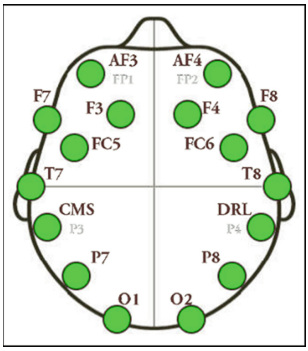

Fig. 1. Position of the electrodes on the EEG headset with labels along the 10–20 system.

Autonomic arousal will be operationalized as skin conductance level measured with disposable electrodes filled with electrically neutral gel and adhered planar surface of the foot. A Biopac MP150 system will be used to collect the skin conductance data at 1000 Hz. Emotional valence will be operationalized as the relative activation of the corrugator supercilli muscle group (facial EMG). Corrugator EMG will be measured using a pair of mini (4 mm) reusable AG/AGCL electrodes filled with electrolyte gel placed above the subject's left eye after dead skin cells has been removed by a skin prep pad containing rubbing alcohol and pumice. The bipolar corrugator measures will be collected using the Biopac MP150 system with high pass filters set at 8 Hz. The full wave signal will be rectified and then contour integrated online at a time constant of 100 ms, and then sampled at 1000 Hz by the Biopac MP150 system.

3.4 Procedures

Participants will complete the experimental procedure individually after providing informed consent approved by the university's Institutional Review Board. The experiment will take place in an individual lab room. The entire research session will take about 120 min.

Participants will be seated in a high back chair to minimize movement. They will be provided a standard keyboard and a mouse. The experimenter will then explain the procedure for attaching the physiological electrodes and fitting the EEG cap, answering any questions posed by the participant. After obtaining adequate impedance readings for the EDA, EMG and EEG measures, the protocol will continue with another brief handedness questionnaire.

Next the participant will watch two videos, one familiarizing them with the rules of the game Kalah and one familiarizing them with the Eclipse environment. They will then be instructed to attempt the practice task, which is changing the capture rule to the empty house capture rule. Upon successful completion of the practice tasks they will receive the next four tasks in a fully counterbalanced format (2 tasks in 2 treatments).

After the tasks are completed, the experimenter will remove the EEG cap and physiological sensors. The participant will then complete the post-experiment questionnaire. Finally, the participant will be debriefed, compensated for their time, and released.

3.5 Data Cleaning and Preparation

EEG data will be cleaned and analyzed using EEGLab [34]. One limitation of EEG is that cortical bioelectrical activity is extremely small in magnitude when compared to muscle movements across the head. Therefore, participant movement introduces artifacts of high-frequency and magnitude into the EEG data. These will be removed using two methods: EEGlab probability calculations and visual inspection. The EEGLab artifact rejection algorithm uses deviations in microvolts greater than three standard deviations from the mean to reject specific trials. However, additional artifacts are also apparent to the trained eye, so visual inspection of trials is essential in artifact removal [34].

In addition to trial-by-trial removal of artifacts, occasionally specific EEG channels must be rejected in an individual subject's data due to unacceptable impedance levels. This can be done in the current study using an automatic impedance detection feature of EEGLab.

Electrodermal and facial EMG data will be aggregated to mean values per second using Biopac's Acqknowledge software. Change scores will be calculated by subtracting the physiological level at the onset of each target statement during the online discussion from each subsequent second across a 6-second window.

3.6 ICA Analysis of EEG Data

The first step of the EEG analysis will be an Independent Components Analysis (ICA) at the individual level. A common problem in neurophysiology research results

from the collection of large amounts of data which, based upon the Central Limit Theorem, become normally distributed. However, the brain is comprised of discrete patches of cortex that are very active at some points in time and relatively non-active at others (i.e. – activity is not normally distributed across the scalp) [35]. ICA overcomes this problem by taking this Gaussian data and rotating it until it becomes non-Gaussian, thereby isolating independent components of activation. The ICAs are distributed patterns of activation across the 14-electrodes in the EEG system.

Initially, an EEGLab ICA performs a Principal Components Analysis (PCA). At each electrode site the program assesses which of the other electrode sites account for the most variance in the signal. Taking these weighted values it then relaxes the orthogonality constraint of PCA to isolate individual components of activation [35]. Each ICA component then represents a pattern of activation over the entire brain, not solely the activity present at a specific electrode. The number of independent components (ICs) depends on the number of electrodes in the dataset, as the algorithm is working in an *N*-dimensional space (where N is the number of electrodes). Most participants in the current study are expected to generate 14 distinct ICs, since our recording device has 14 electrodes.

Finally, using the *K*-means component of EEGlab the independent components at the individual level will be grouped into clusters containing similar components using procedures recommended by [36]. This procedure clusters similar ICs based upon their latency, frequency, amplitude, and scalp distribution [36]. Relevant clusters will be identified and a time-frequency decomposition will be performed to examine changes in event-related desyncronization of the alpha rhythm.

4 Potential Implications

This study seeks to better understand the effects of software design strategies on cognitive load and arousal. The findings will have two strong implications for future research. First, the study will elucidate which design structures create the greatest level of cognitive load and arousal on software engineers, which has significant practical implications for software development. Second, and perhaps more importantly, this study will employ tools from cognitive neuroscience to examine how software engineers react to software design. This introduces what we term Neuro Software Engineering or NeuroSE into the software engineering field. Further research can examine other types of code and design structures and strategies in a similar way as this study to determine optimal design structures and strategies, from a cognitive and emotional standpoint.

Furthermore, the software engineering field can apply these tools to other research and practice areas such as language design and software engineering education. By understanding the cognitive load that various language artifacts and software structures impose on (experienced and novice) developers, we can better design these artifacts and structures, so that the experience of software development and the way that we teach it is better attuned to human cognitive capabilities and limitations.

References

1. Chow, T., Cao, D.-B.: A survey study of critical success factors in agile software projects. J. Syst. Softw. **81**(6), 961–971 (2008)
2. DeMarco, T., Lister, T.: Waltzing with Bears: Managing Risk on Software Projects. Addison-Wesley, Boston (2013)
3. Lawrie, D., Morrell, C., Feild, H., Binkley, D.: What's in a Name? a study of identifiers. In: 14th IEEE International Conference on Program Comprehension (ICPC 2006), pp. 3–12 (2006)
4. Dijkstra, E.W.: Cooperating sequential processes. In: Hansen, P.B. (ed.) The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls, pp. 65–138. Springer, New York (2002)
5. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15**, 1053–1058 (1972)
6. Tempero, E.: An empirical study of unused design decisions in open source java software. In: 2008 15th Asia-Pacific Software Engineering Conference, pp. 33–40 (2008)
7. Gorschek, T., Tempero, E., Angelis, L.: A large-scale empirical study of practitioners' use of object-oriented concepts. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering, pp. 115–124 (2010)
8. Kazman, R., Cai, Y., Mo, R., Feng, Q., Xiao, L., Haziyev, S., Fedak, V., Shapochka, A.: A case study in locating the architectural roots of technical debt. Presented at the Proceedings of the 37th International Conference on Software Engineering, Vol. 2, Florence (2015)
9. Dimoka, A.: What does the brain tell us about trust and distrust? evidence from a functional neuroimaging study. MIS Q. **34**, 373–396 (2010)
10. Minas, R.K., Potter, R.F., Dennis, A.R., Bartelt, V., Bae, S.: Putting on the thinking cap: using neurois to understand information processing biases in virtual teams. J. Manag. Inf. Syst. **30**, 49–82 (2014)
11. Siegmund, J.: Measuring Program Comprehension with Fmri
12. McCabe, T.J.: A complexity measure. IEEE Trans. Softw. Eng. **2**, 308–320 (1976)
13. Halstead, M.H.: Elements of Software Science **7**. Elsevier, New York (1977)
14. Mo, R., Cai, Y., Kazman, R., Xiao, L.: Hotspot patterns: the formal definition and automatic detection of architecture smells. In: 2015 12th Working IEEE/IFIP Conference on Software Architecture, pp. 51–60 (2015)
15. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**, 476–493 (1994)
16. Lorenz, M., Kidd, J.: Object-Oriented Software Metrics: A Practical Guide. Prentice-Hall Inc, Upper Saddle River (1994)
17. e Abreu, F.B.: The mood metrics set. In: Proceedings of ECOOP, p. 267 (1995)
18. Harrison, R., Counsell, S.J., Nithi, R.V.: An evaluation of the mood set of object-oriented software metrics. IEEE Trans. Softw. Eng. **24**, 491–496 (1998)
19. Lang, J.E., Bogovich, B.R., Barry, S.C., Durkin, B.G., Katchmar, M.R., Kelly, J.H., McCollum, J.M., Potts, M.: Object-oriented programming and design patterns. ACM SIGCSE Bull. **33**, 68–70 (2001)
20. Paas, F., Renkl, A., Sweller, J.: Cognitive load theory and instructional design: recent developments. Educ. Psychol. **38**, 1–4 (2003)
21. Baddeley, A.: Working memory. Science **255**, 556–559 (1992)
22. Conway, A.R.A., Engle, R.W.: Working memory and retrieval: a resource-dependent inhibition model. J. Exp. Psychol. Gen. **123**, 354–373 (1994)

23. Welsh, M.C., Satterlee-Cartmell, T., Stine, M.: Towers of hanoi and london: contribution of working memory and inhibition to performance. Brain Cogn. **41**, 231–242 (1999)
24. D'Esposito, M.: From cognitive to neural models of working memory. Philos. Trans. Royal Soc. B Biol. Sci. **362**, 761–772 (2007)
25. D'Esposito, M., Detre, J.A., Alsop, D.C., Shin, R.K., Atlas, S., Grossman, M.: The neural basis of the central executive system of working memory. Nature **378**, 279–281 (1995)
26. Wager, T.D., Jonides, J., Reading, S.: Neuroimaging studies of shifting attention: a meta-analysis. NeuroImage **22**, 1679–1693 (2004)
27. Gevins, A., Smith, M.E., McEvoy, L., Yu, D.: High-resolution EEG mapping of cortical activation related to working memory: effects of task difficulty, type of processing, and practice. Cereb. Cortex **7**, 374–385 (1997)
28. Lavie, N.: Distracted and confused? selective attention under load. Trends Cogn. Sci. **9**, 75–82 (2005)
29. Shi, Y., Ruiz, N., Taib, R., Choi, E., Chen, F.: Galvanic skin response (Gsr) as an index of cognitive load. Presented at the CHI 2007 Extended Abstracts on Human Factors in Computing Systems. San Jose (2007)
30. Mehler, B., Reimer, B., Coughlin, J., Dusek, J.: Impact of incremental increases in cognitive workload on physiological arousal and performance in young adult drivers. Transp. Res. Rec. J. Transp. Res. Board **2138**, 6–12 (2009)
31. Moody, D.: The "Physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**, 756–779 (2009)
32. Sutcliffe, A.G., Maiden, N.A.M.: Analysing the novice analyst: cognitive models in software engineering. Int. J. Man-Mach. Stud. **36**, 719–740 (1992)
33. Herwig, U., Satrapi, P., Schönfeldt-Lecuona, C.: Using the international 10–20 EEG system for positioning of transcranial magnetic stimulation. Brain Topogr. **16**, 95–99 (2003)
34. Delorme, A., Makeig, S.: EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J. Neurosci. Methods **134**, 9–21 (2004)
35. Onton, J., Makeig, S.: Information-based modeling of event-related brain dynamics. In: Christa, N., Wolfgang, K. (Eds.) Progress in Brain Research, vol. 159, pp. 99–120. Elsevier, Amsterdam (2006)
36. Delorme, A., Makeig, S.: EEGLAB Wikitutorial, 12 May. http://sccn.ucsd.edu/wiki/PDF:EEGLAB_Wiki_Tutorial