# CODERSMUSE: Multi-Modal Data Exploration of Program-Comprehension Experiments

Norman Peitek
*Leibniz Institute for Neurobiology*
Magdeburg, Germany

Sven Apel
*University of Passau*
Passau, Germany

André Brechmann
*Leibniz Institute for Neurobiology*
Magdeburg, Germany

Chris Parnin
*NC State University*
Raleigh, North Carolina, USA

Janet Siegmund
*University of Passau*
Passau, Germany

***Abstract*—Program comprehension is a central cognitive process in programming. It has been in the focus of researchers for decades, but is still not thoroughly unraveled. Multi-modal psycho-physiological and neurobiological measurement methods have proved successful to gain a more holistic understanding of program comprehension. However, there is no proper tool support that lets researchers explore synchronized, conjoint multi-modal data, specifically designed for the needs in program-comprehension research. In this paper, we present CODERSMUSE, a prototype implementation that aims to satisfy this crucial need.**

***Index Terms*—program comprehension, data exploration, functional magnetic resonance imaging**

## I. INTRODUCTION

Program comprehension is the fundamental process of understanding program code. Understanding program comprehension is important, because programmers spend most of their time maintaining—and therefore comprehending—existing program code [15].

Despite decades of research, our understanding of how programmers comprehend code is still limited. Many questions, such as *what makes a great programmer* or *how to teach novices*, are still unanswered. To shed light onto the underlying cognitive processes of program comprehension, researchers have begun to adopt psycho-physiological and neurobiological measurement methods [12].

Each method provides distinct insights into certain aspects of program comprehension, while neglecting others. For example, eye-tracking and behavioral data (e.g., response time and correctness) let us observe a programmer's strategy to solve a comprehension task, but provide only little information on a programmer's stress level during a task. To measure the stress level, we need psycho-physiological methods, such as heart rate variability or electrodermal activity [7], which, however, do not provide information on a programmer's comprehension strategy. Finally, neuroimaging methods allow us to study underlying cognitive processes [14], but not necessarily reveal a programmer's specific strategy to solve a comprehension task.

Basically, each method comes with inherent benefits and shortcomings. Hence, observing programmers with multiple measurement methods simultaneously offers a way to a better understanding of the complex behavioral and cognitive processes of program comprehension [10], [11]. If we are able to combine multiple measurement methods and integrate the collected data in a meaningful way, we have a powerful framework to understand even subtle phenomena in program comprehension, such as the effect of identifier naming [5].

Integrating multiple data streams is challenging, especially when they have different characteristics. For example, eye tracking provides one-dimensional data with high temporal resolution (in the order of milliseconds). By contrast, neuroimaging methods, such as functional magnetic resonance imaging (fMRI), provide data that are three dimensional and have a low temporal resolution, in the order of hundreds of milliseconds (cf. Section II-A3). To increase our insights from multi-modal experiments, we need appropriate techniques to properly integrate such different data streams [10].

We have been developing concepts to conjointly analyze multi-modal experiment data and implement them in our tool CODERSMUSE. We are following a rapid prototyping approach, so we are considering four data streams for now, that is, behavioral, eye-tracking, psycho-physiological, and fMRI data. We selected these methods, because they are the most common methods [2], [6], [13], [14].

The prototype is designed to analyze and explore all data streams synchronously. For example, in a recent experiment, we investigated the neural efficiency of top-down comprehension [11], [14], in which we asked participants to comprehend code snippets inside an fMRI scanner. In addition to brain activation, we also collected behavioral data (i.e., efficiency and correctness), eye-tracking data, and psycho-physiological data (i.e., heart rate, respiration). With CODERSMUSE, we can jointly explore all data streams instead of having to rely on individual tools and analysis for each data stream. This way, CODERSMUSE enables us to generate substantially new and more holistic hypotheses for follow-up studies of program comprehension.

TABLE I
ALL DATA STREAMS SUPPORTED IN CODERSMUSE AND THEIR CHARACTERISTICS, TYPICAL PREPROCESSING, AND MEASUREMENTS.

| Data | Type | Delay | Temporal Resolution | Typical Preprocessing | Typical Measurement |
|---|---|---|---|---|---|
| | *Important for visualization* | *Important for data synchronization* | | *Done before import to* CODERSMUSE | |
| Behavioral | Individual events | None | n/a | Coding | Response time, correctness |
| Eye-tracking | Data stream | None | Down to less than milliseconds | Smoothing, saccade, and fixation detection | Eye gaze |
| Physiological | Data streams | Few seconds | Typically milliseconds | Smoothing | Stress level |
| fMRI | 3-dimensional set of data streams | Several seconds | Typically a second | 3D-motion correction, slice-scan-time correction, temporal filtering, spatial smoothing | Brain activation |


Fig. 1. Screenshot of CODERSMUSE. On the top right, the user can select a specific task (2). A time slider allows the user to explore data across a task's time line (1). Then, several data streams will be displayed: eye-tracking data (3), behavioral data (4), psycho-physiological data (5), and fMRI data (6) (7).

## II. PROTOTYPE IMPLEMENTATION OF CODERSMUSE

Along with this paper, we publish an open-source prototype implementation of CODERSMUSE. We provide a demo video and an open-source version with sample data on the project's Web site.[1]

In Figure 1, we show a screenshot with annotated feature explanations, which are numbered in yellow circles and which we explain in detail in this section.

### A. Integrated Modalities

Currently, CODERSMUSE supports four different modalities, of which we provide an overview in Table I.

*1) Behavioral Data (4):* Behavioral data include events derived from a participant's actions, including response correctness, response time, and click duration (i.e., time between button-down and button-up events).

*2) Eye-Tracking Data (3):* Eye-tracking data are visualized on top of an image of the current task (e.g., a code snippet). We currently support static images, which is the most common type in eye-tracking experiments. Dynamic content (such as scrolling on a Web site) is not supported but may be added in the future. Eye-tracking data are visualized as a *scanpath* [8]. Fixations and saccades are highlighted with different colors. Other visualizations, such as heatmaps, can be added in the future.

*3) fMRI Data:* fMRI data provide insights into the underlying cognitive processes during program comprehension. CODERSMUSE supports two visualizations of fMRI data. First (6), it shows the brain activation strength over time for a specific brain region of interest (e.g., to observe working memory load during a task). Second (7), CODERSMUSE visualizes the full-brain activation with the Python library *NIPY*[2]

[1] https://github.com/brains-on-code/CodersMUSE/

[2] http://nipy.org/nipy/

to observe higher level patterns (e.g., to identify involvement of a brain area at a specific time).

*4) Psycho-Physiological Data (5):*
Last, CODERSMUSE visualizes heart rate variability, respiration, and electrodermal activity in the form of numeric time series.

### B. Features and Challenges

The key feature of exploring data with CODERSMUSE is essentially a real-time replay of collected data (1). That is, users can *(re)play* data of an experiment session and simultaneously observe all data streams. The user can also use the time slider to dynamically move through a task's time line to examine an event more closely (1). In Figure 1, we set CODERSMUSE to show the data of an experiment that are split into individual tasks. That is, we select a specific task from the entire experiment and show the data of a specific task (2). CODERSMUSE may also show the entire data set of all tasks, but this limits the usefulness of the eye-tracking-data view (as the presented code typically changes with each task).

The complexity of CODERSMUSE stems from the inherent differences in the characteristics of the modalities, which poses major challenges for a proper conjoint exploration (cf. Table I).

*1) Data Preprocessing:* Data preprocessing is an important step to ensure high data quality, which is a prerequisite to obtain meaningful insights. But the mandatory preprocessing varies between modalities. For example, eye-tracking data require fundamentally different preprocessing than fMRI data (cf. Table I).

A re-implementation of every necessary preprocessing step for each modality would be inefficient and error-prone. Thus, the current prototype of CODERSMUSE relies on already preprocessed data. Users are required to preprocess their data before importing them into CODERSMUSE. On the CODERSMUSE's Web site, we provide template scripts and guides on how to integrate your own data, including the necessary preprocessing.

*2) Data Synchronization:* Another challenge of CODERSMUSE is to properly synchronize the timing of each data visualization. The integrated modalities exhibit different temporal delays. For example, fMRI measures the biological effect of cognitive processes (i.e., haemodynamic response), which means that the data stream is delayed by around five seconds [3]. To counteract this effect, the displayed fMRI data are offset by six seconds. Similarly, the response of electrodermal activity is typically delayed by about two seconds [1], so we offset the presented time frame by the same amount. Eye-tracking data have no delay. All offsets are default settings and can be configured.

*3) Data Visualization:* Each modality provides fundamentally different data, so we need different visualizations. For example, visualizing one-dimensional eye-tracking data is different than visualizing three-dimensional neuroimaging data. Thus, each data stream implements its own visualization, which is inspired by state-of-the-art tools.

### C. Implementation

Due to the performance requirements of handling the wealth of data, CODERSMUSE's prototype is implemented as a desktop program. It is based on Python 3.6 and Qt 5.11, making it available cross-platform.

CODERSMUSE follows a plug-in architecture, such that each modality is implemented as a separate plug-in. This way, different modalities can be easily supported: For example, researchers can swap the fMRI plug-in with a new plug-in (e.g., for fNIRS, currently not implemented). Furthermore, each plug-in can be further enhanced to individual needs, for example, with additional view options. This way, CODERSMUSE is also customizable.

Each plug-in creates a view for its respective data set. When the user explores data along the time slider, CODERSMUSE's core triggers a view update with the current time stamp to each plug-in. When the user interacts with a specific view (e.g., to change observed position in the brain), the respective plug-in accepts the request and renews the view. An interaction between plug-ins is currently not supported.

## III. FUTURE WORK

Due to the complexity and novelty of this endeavor, there is still substantial work left. We share our early version to enable the community, which is starting to embrace multi-modal program-comprehension experiments, to shape the further development of CODERSMUSE. Depending on the study, each researcher may have different use cases for CODERSMUSE, which we invite to communicate, for example, on the project's Web site or directly to us.[1] For our own purposes, we foresee the inevitable need for extension, which we discuss next.

### A. Additional Modalities

So far, we have focused on supporting fMRI as neuroimaging method, but there are alternatives, such as functional near-infrared spectroscopy (fNIRS) [5]. fNIRS does not require such a physically restrained setting as fMRI. fNIRS measures the same underlying biological effect as fMRI, and therefore fNIRS data are similarly delayed as fMRI data. However, fNIRS does not provide a three-dimensional data set (in the order of about 100,000 time series), but instead aggregates the measured brain activation into a handful of data streams. To support fNIRS data in CODERSMUSE, we intend to develop an according plug-in and extend the guidelines to describe how to use this plug-in.

Another extension would be to integrate electroencephalography (EEG) data, for example, to observe cognitive load of programmers, as done by Crk et al. [4], which can also be recorded simultaneously with fMRI data. EEG data differ from fMRI data in that they are not delayed, have a higher temporal resolution (in the order of milliseconds), and provide typically 64 data streams, collected via channels located at different positions around the skull.

Of course, increasing the amount of data might also affect the performance of CODERSMUSE, such that we shall further refine the underlying data-processing architecture.

### B. Data Annotation and High-Level Patterns

To deal with the wealth of data obtained from a multi-modal experiment, researchers should be able to annotate individual events in each individual data stream or across data streams, such as a peak in the neuronal response. At first, this may merely be a manual process, but could be extended by automatic techniques (e.g., based on a classifier) to detect similar events in other parts of an experiment (e.g., as ATLAS is offering [9]). Eventually, we aim at extracting higher level patterns across data streams. For example, CODERSMUSE may indicate that long fixations on loop initializations (eye tracking) trigger an activation in working memory (neuroimaging) and increased cognitive load (psycho-physiological data). Such extracted high level patterns would allow researchers to generate new hypotheses for follow-up studies and eventually to a more holistic understanding of program comprehension.

## IV. RELATED WORK

A meaningful combination of neuroimaging and eye-tracking data, in addition to other modalities, is still in its infancy, not only in program-comprehension research, but also in neuroscience. We are not aware of any commercial or scientific tool that integrates several modalities in one offline data-exploration tool that fits our needs. ATLAS is a multi-modal data-annotation tool [9], but it does not offer a convenient integration of eye-tracking data, which is essential for understanding a programmer in action [11].

When analyzing single-modality data, numerous tools exist. For example, Ogama is an open-source tool to record, explore, and analyze eye-tracking data [16]. BrainVoyager™[3] and SPM[4] are two tools to analyze fMRI data. While none of these tools combine different data streams, they inspired individual views of CODERSMUSE (e.g., visualizations for the eye-tracking data are inspired by Ogama).

## V. CONCLUSION

CODERSMUSEenables researchers to explore synchronized and integrated multi-modal data. This way, we intend to unravel the mysteries of program comprehension, which has been possible only to a limited degree, so far. In the future and with the input of the community, we will mature CODERSMUSE, thereby making the analysis of multi-modal experiments accessible to a wide range of users.

## ACKNOWLEDGMENTS

Brechmann's and Siegmund's work is supported by DFG grants BR 2267/7-1 and SI 2045/2-1. Siegmund's work is further funded by the Bavarian State Ministry of Education, Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B). Parnin's work is supported by the National Science Foundation under grant number 1755762.

[3]Brain Innovation B.V., Netherlands, brainvoyager.com
[4]https://www.fil.ion.ucl.ac.uk/spm/

## REFERENCES

[1] W. Boucsein. *Electrodermal Activity*. Springer Science & Business Media, 2012.

[2] J. Castelhano, I. Duarte, C. Ferreira, J. Duraes, H. Madeira, and M. Castelo-Branco. The Role of the Insula in Intuitive Expert Bug Detection in Computer Code: An fMRI Study. *Brain Imaging and Behavior*, pages 1 – 15, 2018.

[3] B. Chance, Z. Zhuang, C. UnAh, C. Alter, and L. L. Cognition-Activated Low-Frequency Modulation of Light Absorption in Human Brain. *Proc. Nat'l Academy Sciences of the United States of America (PNAS)*, 90(8):3770–3774, 1993.

[4] I. Crk, T. Kluthe, and A. Stefik. Understanding programming expertise: An empirical study of phasic brain wave changes. *ACM Trans. Comput.-Hum. Interact.*, 23(1):2:1–2:29, 2015.

[5] S. Fakhoury, Y. Ma, V. Arnaoudova, and O. Adesope. The effect of poor source code lexicon and readability on developers' cognitive load. In *Proc. Int'l Conf. Program Comprehension (ICPC)*, page 11 pages. IEEE, 2018.

[6] B. Floyd, T. Santander, and W. Weimer. Decoding the Representation of Code in the Brain: An fMRI Study of Code Review and Expertise. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 175–186. IEEE, 2017.

[7] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger. Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 402–413. ACM, 2014.

[8] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. OUP Oxford, 2011.

[9] S. Meudt, L. Bigalke, and F. Schwenker. Atlas – annotation tool using partially supervised learning and multi-view co-learning in human-computer-interaction scenarios. In *Int'l Conf. Information Science, Signal Processing and their Applications (ISSPA)*, pages 1309–1312. IEEE, 2012.

[10] N. Peitek, J. Siegmund, C. Parnin, S. Apel, and A. Brechmann. Toward Conjoint Analysis of Simultaneous Eye-Tracking and fMRI Data for Program-Comprehension Studies. In *Proc. Int'l Workshop on Eye Movements in Programming (EMIP)*, pages 1:1–1:5. ACM, 2018.

[11] N. Peitek, J. Siegmund, C. Parnin, S. Apel, J. Hofmeister, and A. Brechmann. Simultaneous Measurement of Program Comprehension with fMRI and Eye Tracking: A Case Study. In *Proc. Int'l Symposium Empirical Software Engineering and Measurement (ESEM)*, pages 24:1–24:10. ACM, 2018.

[12] J. Siegmund. Program Comprehension: Past, Present, and Future. In *Int'l Conf. Software Analysis, Evolution, and Reengineering (SANER)*, pages 13–20. IEEE, 2016.

[13] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann. Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 378–389. ACM, 2014.

[14] J. Siegmund, N. Peitek, C. Parnin, S. Apel, J. Hofmeister, C. Kästner, A. Begel, A. Bethmann, and A. Brechmann. Measuring Neural Efficiency of Program Comprehension. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*, pages 140–150. ACM, 2017.

[15] R. Tiarks. What Programmers Really Do: An Observational Study. *Softwaretechnik-Trends*, 31(2):36–37, 2011.

[16] A. Voßkühler, V. Nordmeier, L. Kuchinke, and A. M. Jacobs. Ogama (open gaze and mouse analyzer): Open-source software designed to analyze eye and mouse movements in slideshow study designs. *Behavior Research Methods*, 40(4):1150–1162, 2008.