# 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 and has been in the focus of researchers for decades, but is still not thoroughly unraveled. Multi-modal measurement methods are a way 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 software engineering. 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 [14].

However, 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 neuroimaging methods [11].

Each method provides interesting insights into some 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 [6], which, however, do not provide information on a programmer's strategy to solve a comprehension task. Finally, neuroimaging methods allow us to study underlying cognitive processes [13], but not necessarily reveal a programmer's specific strategy to solve a comprehension task.

Basically, each method comes with inherent benefits and drawbacks. Hence, observing programmers with multiple measurement methods simultaneously offers a way to a better understanding of the complex underlying cognitive processes of program comprehension [9], [10]. 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 [4].

However, 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), 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 approaches to properly integrate such different data streams [9].

To this end, we are 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 focusing on four data streams for now, that is, behavioral, eye-tracking, psycho-physiological, and fMRI data. We selected these methods, because they are the most promising methods and are widely used [5], [12], [13].

The prototype is designed to analyze and explore all data streams with a systematic approach. For example, in a recent experiment, we investigated the neural efficiency of top-down comprehension [10], [13], in which we asked participants to comprehend code snippets inside an fMRI scanner. In addition to their brain activation, we also collected behavioral data (i.e., efficiency and correctness), eye-tracking data, and psycho-physiological data (i.e., heart rate). CODERSMUSE allows users to 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.

## II. PROTOTYPE IMPLEMENTATION OF CODERSMUSE

Along with this paper, we publish an open-source prototype implementation of CODERSMUSE. We provide a demo video,

TABLE I
ALL DATA STREAMS, WHICH ARE 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 | Up 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).

an executable version with sample data, and guides on how to explore your own data set 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. In essence, CODERSMUSE provides a synchronized, integrated presentation of different modalities.

### A. Integrated Modalities

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

*1) Behavioral Data (4):* The behavioral data are 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):* The 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. The eye-tracking data are drawn as a *scanpath* [7]. Fixations and saccades are highlighted with different colors. Other visualizations, such as heatmaps, could also 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 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] to observe

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

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

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

*4) Psycho-Physiological Data (5):* Lastly, CODERSMUSE visualizes heart rate variability, respiration, and electrodermal activity, which are numeric time series.

### B. Features and Challenges

The key feature to explore 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 task. That is, we select a specific task from the entire experiment and will be able to show the data of a specific task (2). CODERSMUSE can 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 characteristics between 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 necessary 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.[1]

*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 [2]. To counteract this effect, the displayed fMRI data are offset by five 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 to handle 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. In this way, CODERSMUSE is also customizable.

Each plug-in creates a view for their 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 extensions, 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) [4]. 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. [3], 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 not just one, but 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 need to further refine the underlying data-processing architecture.

### B. Data Annotation

To deal with the wealth of data of multi-modal experiments, 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 approaches (e.g., based on a classifier) to detect similar events in other parts of an experiment (e.g., as ATLAS is offering [8]). 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 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. Use Cases

Analyzing empirical studies of program comprehension, particularly multi-modal ones, is time-consuming. The goal of CODERSMUSE is to not only compute numerical results for a paper, but to support insightful data exploration, which is necessary to build a holistic understanding of program comprehension. By exploring data, we can build new hypotheses to be tested in follow-up studies. CODERSMUSE aids both, discovery and testing of hypotheses.

Users can investigate the behavior (behavioral and eye-tracking data) as well as measurements of program comprehension (neuroimaging, psycho-physiological) for each individual participant. The real-time replay of data lets users dive into and, hopefully, understand detailed events during an experiment. CODERSMUSE's views are designed to provide a comprehensive observation of each data stream.

## V. Related Work

A meaningful combination of neuroimaging and eye-tracking data, in addition to other modalities, is still in its infancy, not only in software-engineering research, but also in neuroscience. We are not aware of a commercial or scientific tool that integrates many modalities in one offline data-exploration tool that fits our needs. ATLAS is a multi-modal data-annotation tool [8], but does not offer a convenient integration of eye-tracking data, which is essential for understanding a programmer [10].

When analyzing single-modality data, numerous tools exist. For example, Ogama is an open-source tool to record, explore, and analyze eye-tracking data [15]. 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).

[3]Brain Innovation B.V., Netherlands, brainvoyager.com

[4]https://www.fil.ion.ucl.ac.uk/spm/

## VI. Conclusion

We presented the tool CODERSMUSE, which enables 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 hope to mature CODERSMUSE, thereby making the analysis of multi-modal experiments accessible to a wide range of users.

## References


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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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, July 2012.

[9] 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*, pages 1:1–1:5. ACM, 2018.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.