A Replication Study on Code Comprehension and Expertise using Lightweight Biometric Sensors

Davide Fucci*, Daniela Girardi†, Nicole Novielli†, Luigi Quaranta†, Filippo Lanubile†
*University of Hamburg, Germany, fucci@informatik.uni-hamburg.de
†University of Bari, Italy, {name.surname}@uniba.it

***Abstract*—Code comprehension has been recently investigated from physiological and cognitive perspectives through the use of medical imaging. Floyd *et al.* (i.e., the original study) used fMRI to classify the type of comprehension tasks performed by developers and relate such results to their expertise. We replicate the original study using lightweight biometrics sensors which participants (28 undergrads in computer science) wore when performing comprehension tasks on source code and natural language prose. We developed machine learning models to automatically identify what kind of tasks developers are working on leveraging their brain-, heart-, and skin-related signals. The best improvement over the original study performance is achieved using solely the heart signal obtained through a single device (BAC 87% vs. 79.1%). Differently from the original study, we were not able to observe a correlation between the participants' expertise and the classifier performance (τ = 0.16, p = 0.31). Our findings show that lightweight biometric sensors can be used to accurately recognize comprehension tasks opening interesting scenarios for research and practice.**

***Index Terms*—software development tasks, biometric sensors, machine learning.**

I. Introduction

Developers spend more time understanding code than for any other activities [1], [2]. Code comprehension is crucial for several software development activities, such as code review [3], [4], [5]. As such, researchers have developed several strategies to study this activity [6], [7].

Despite its importance, we only have an initial grasp about the role that human physiological factors [8] play for code comprehension. A first step is to understand the relationship between code comprehension and the underlying cognitive mechanisms is studying to what extent biometric feedback can help discriminating between code and other comprehension tasks—e.g., natural language comprehension. To that end, Floyd et al. [9] (i.e., the original study replicated in this paper) use functional Magnetic Resonance Imaging (fMRI) to build a classifier able to distinguish these two tasks based on brain activity. However, their approach, on top of being expensive (approximately $500/hour [9]), limits the ecological validity of the results. In this paper, we study to what extent we can replicate the results of the original study using lightweight biometric sensors.

The choice of using lightweight biometric sensors—i.e., non-intrusive, wearable, and affordable devices—to measure human physiology is supported by the results of recent research demonstrating its potential application to software engineering. For example, cognitive-aware IDEs can support developers for code comprehension tasks (e.g., during code-review [10]) and for fostering their productivity by monitoring interruptibility [11]. Similarly, by leveraging biometric information, IDEs can automatically adapt to a development specific tasks. Fritz et al. [12] uses a combination of eye-tracker, electrodermal activity, and electroencephalography sensors to measure the difficulty of a development task. Fakhoury et al. [13] used fNIRS (a brain imaging technique which uses sensors connected to a portable headband) to study the effects of poor code readability and lexicon on novice developers' code comprehension.

In this study, we use *electroencephalogram* (EEG), *electrodermal* (EDA), and *heart-related* sensors to replicate the original study. Although such biometrics cannot give a detailed account of a developer's cerebral activity (e.g., activated brain areas) as in the case of fMRI, they can be used to sense variation in cognitive load [14], [15] associated with a cognitive task. We developed machine learning models that use features extracted from biometric signals collected from 28 participants comprehending source code and natural language. Our approach outperforms the one presented in the original study while achieving its best results using only features from physiological signals acquired using a single wearable device.

The contributions of this paper are:

- an approach for automatic recognition of two comprehension tasks that leverages lightweight biometrical sensors;
- an empirical investigation of which combination of physiological sensors and measurements are most effective at predicting a code comprehension task;
- a replication package including the material to replicate our experiment including the datasets, scripts, and benchmarks to reproduce and evaluate the machine learning models.[1]

This paper is organized according to standard replication report guidelines for software engineering studies [16]. We consider the current study an external, independent, and differentiated replication [17], [18] as a completely independent set of researchers replicated the original study while making intentional changes to it.

Paper organization. Section II reviews the existing literature regarding the use of biometrics. Section III summarizes the original study goals, its settings, and results. The main changes to the original study and the details about this replica-

[1]https://github.com/collab-uniba/Replication_Package_ICPC

tion are reported in Section IV while our machine learning approach is described in Section V. Section VI reports the results of this replication while Section VII summarizes its limitation and implications. Finally, we conclude in Section VIII.

II. Use of Biometrics in Software Engineering

The software engineering research community has studied the relationship between the developers' cognitive state—measured using physiological signals—and several aspects of software development, like code comprehension [19], productivity [20], and software quality [21].

Parnin [22] used sub-vocal utterances, emitted by software developers, to study the complexity of two programming task. The author used an electromyogram (EMG) to show that those signals are correlated to the cognitive patterns that developers follow when tackling a programming task. Fritz et al. [12] combined three physiological features (i.e., eye movement, electrical signal of skin and brain) with a similar goal. The authors showed that the three biometrics together provide the best combination for predicting the difficulty of a task (84.38% precision, and 69.79% recall). Moreover, they demonstrated that off-the-shelf devices can be used to build accurate, online classifiers of difficult code chunks.

Developers productivity has recently been the subject of research exploiting physiological measures. For example, Radevski et al. [20] proposed a framework for continuous monitoring of developers' productivity based on brain electrical activities. Müller and Fritz [21] used an ensemble of biometrics to measure the progress and interruptibility of developers performing small development tasks. They demonstrated that it is possible to classify the emotions experienced by developers while working on a programming task using biometrics (i.e., brainwave frequency, pupil size, heart rate) with an accuracy of 71%. The progress experienced by developers was predicted at a similar rate, but using a different set of biometrics (i.e., EDA signal, skin temperature, brainwave frequency, and the pupil size).

Müller and Fritz [23] investigated the use of physiological measures for real-time identification of code quality concerns in a real-world setting. Using heart-, skin-, and brain-related biometrics, the authors identified difficult parts of the system—e.g., low-quality code containing bugs. The authors provide some evidence that biometrics can outperform traditional code-related metrics to identify quality issues in a code base.

For Code Comprehension. As far as code comprehension is concerned, two similar studies, Siegmund et al. [8] and Ikutani and Uwano [24], assessed the brain activity of developers involved in code comprehension tasks. Siegmund et al. [8] used fMRI to show clear activation patterns in five regions of the brain all related to language processing, working memory, and attention. The study by Ikutani and Uwano [24] uses near-infrared spectroscopy to show that different parts of the brain are activated during code comprehension with respect to a specific sub-task. For example, they distinguish between the areas activated by the workload necessary to memorize a variable and the ones activated by arithmetic calculation.

More recently, Peitek et al. [19] used fMRI to monitor the brain activity of 28 participants involved in the comprehension of 12 source code snippets. Their results show that distinct areas of the brain are activated during such a task. Moreover, the activation patterns suggest that natural language processing is essential for code comprehension. To get a more comprehensive view of the strategies adopted by developers when comprehending source code, Peitek et al. [25] obtained simultaneous measurements of fMRI and eye-tracking devices. They showed strong activation of specific brain areas when code beacons are available. However, their setup was subject to data loss—complete fMRI and eye-tracking data could be collected for 10 out of the 22 participants.

III. Original Study

This section summarizes the original study, giving an overview of its settings, methodology, and results.

A. Research Question

The original study explored the use of fMRI and machine learning techniques to automatically distinguish between code comprehension, code review, and prose review tasks. Moreover, it investigated whether the neural representation of programming and natural languages changes depending on the developer's expertise.

To guide their research, the authors formulated the following research questions [9]:

- *RQ1* - Can we classify which task a participant is undertaking based on patterns of brain activation?
- *RQ2* - Can we relate tasks to brain regions?
- *RQ3* - Can we relate expertise to classification accuracy?

B. Participants and Context

The original study involved 29 students (18 men, 11 women) at the University of Virginia (USA) with basic experience in the C programming language. Among them, two were computer science graduate students, nine were undergraduates in the College of Arts and Sciences, and 18 were undergraduates in the College of Engineering. All participants were right-handed native English speakers, had normal or corrected-to-normal vision, and reported no history of neuropsychological disorders. The authors rewarded the students for their participation with monetary compensation and extra university credits. The experiment was conducted at the University of Virginia.

C. Artifacts

The authors prepared three types of artifacts according to the experimental tasks (see Section III-D).

1) Code snippets with related software maintenance questions.
2) Patches from GitHub Pull Request including code diff and comments.
3) English texts with simple editing request markup.

The artifacts are available at the original experiment website.[2]

[2]https://web.eecs.umich.edu/~weimerw/fmri.html

D. Design

The experiment consisted of three tasks, code comprehension, code Review, and prose review. The tasks were presented as visual stimuli on a special screen installed in the fMRI scanner. Before beginning the experiment, participants signed a consent form for personal data treatment. The original experiment started by showing the participants an instructional video explaining the goal and the different steps of the study. The participants entered the fMRI scanner for an initial anatomical scan. Then, they performed the experimental tasks consisting of four 11-minute sessions where blocks of code review, code comprehension, and prose review were presented in a quasi-random order. Code comprehension and code review blocks contained three tasks each, whereas prose review blocks were composed of six tasks. Tasks containing source code were displayed for 60 seconds and the ones containing prose for 30 seconds. The participants provided their answer (i.e., accept or reject) through an fMRI-compatible button. They were encouraged to respond as quickly and accurately as possible within the allotted time. Between tasks, the screen displayed a fixation cross for an random interval between two and eight seconds. The sessions were completed without interruptions.

E. Summary of Results

The original study authors found that neural representations of programming languages and natural language are distinct. Specifically, they used Gaussian Process Classification to distinguish between code and prose tasks, achieving a balanced accuracy (BAC) of 79%. They show that neural activity in the prefrontal regions strongly drives this distinction. However, their approach performance was lower (BAC = 62%) when comparing code comprehensions to code review, revealing that these tasks are less distinguishable at a neural level.

Finally, authors showed a negative correlation between their classifier performance and the participants' expertise (r = 0.44, p = 0.16), indicating that for experts the neural representation of source code and prose are similar.

IV. Our Study

This section summarizes our replication.

A. Motivation for conducting the replication

We conducted this replication to broaden the original study results by replacing the observed signal (i.e., neural activity sensed through fMRI) with a different set of signals capturing the same construct (i.e., cognitive effort). Moreover, we want to increase the ecological validity of the original study by using sensing devices which can be used in real-world settings.

B. Level of interaction with original experimenters

The authors of the original study did not take part in the replication process; therefore, this replication is to be considered external [17].

We reused a subset of 18 source code snippets that the authors of the original study made available in their replication package.

TABLE I: Settings comparison between the original study and this replication.

Setting	Study	
	Original study	This replication
Experiment site	Univ. of Virginia (USA)	Univ. of Bari (Italy)
# Participants	29	28
Participants experience	Grads and undergrads	Undergrads
# Task	36 tasks four 11-minute sessions	27 tasks three 6-minute sessions
Task type	Code Comprehension Code Review Prose Review	Code Comprehension Prose Comprehension
Physiological signal	Neural	Neural Skin Heart
Physiolocial measure	BOLD	EEG EDA BVP, HR, HRV
Device	fMRI scanner	BrainLink headset Empatica wristband
Classifier	Gaussian Process	Machine Learning (8 algorithms)
Classifier validation	LORO-CV	LORO-CV Hold-out
Classifier metric	Balanced accuracy (BAC)	Balanced accuracy (BAC)

C. Changes to the original experiment

This replication makes explicit changes to the original study.

1) Adaptation of the research question;
2) Partial modification of task presented to the participants through visual stimuli;
3) Different physiological signals captured from the participants performing the task;
4) Modifications to the experimental protocol;
5) Additional machine learning settings.

Table I compares the original study settings to this replication.

Research Questions. In our study, we answer the following research questions:

- RQ_{Clf} - Can we classify which task a participant is undertaking based on *signals collected from lightweight biometric sensors?*
- RQ_{Exp} - Can we relate expertise to classification accuracy?

Our research questions are adapted from those addressed in the original study (see Section III). Specifically, RQ_{Clf} is adapted from RQ1 of the original study, which we modify by including consideration of lightweight biometric sensors instead of fMRI.

In the original study, RQ2 investigates whether the tasks are associated with the activation of specific brain areas (i.e., *Can we relate tasks to brain regions?*). In our study, it is not possible to address this question. The lightweight EEG device we use in our replication to obtain brain-related signals is not capable of registering activation of brain areas as it allows to only collect the signal from the frontal part of the brain. Therefore, we decided to discard the original study RQ2 from our replication. Finally, we address RQ3 from the original study considering the accuracy of each participant best classifiers trained using biometrical signals.

Tasks. In our study, the participants are required to solve a series of code comprehension tasks (see Fig. 1a) and a series of prose comprehension task (see Fig. 1b). As opposed to the original study, we decided to focus only on comprehension; thus, excluding code review and prose review tasks.

Initially, we conducted a pilot study to validate the feasibility of all the original study tasks, including code review. The participants involved in the pilot (i.e., a Ph.D. student and a researcher in Computer Science) perceived code review tasks as too difficult and the entire experiment as too demanding given the allotted time. The pilot participants reported that they felt overwhelmed when performing the code review task and that they ended up providing random answers without actually trying to solve the task. As this represents a threat to construct validity of our study, we discarded the code review tasks.

Finally, to be consistent with the type of activities to compare, we replaced the prose review appearing in the original study with a new prose comprehension task. We operationalized prose comprehension using standard evaluation exercises for high school students (See Fig. 1b). We repeated the pilot study with the same participants, who agreed with the changes.

Physiological signals. In the original study, the authors used images captured from functional magnetic resonance (fMRI) to build a classifier able to distinguish between the tasks a participant is performing. The fMRI provides indirect estimates of brain activity by measuring metabolic changes in blood flow and oxygen consumption. Although this technique allows to understand how the human brain processes software engineering tasks, it is expensive and cannot be used in real-world settings—i.e., to monitor a developer's cognitive activity during daily programming tasks.

Thus, we decided to measure other physiological signals which can be recorded using low cost, lightweight biometric sensors. In our study, we use the BrainLink headset (Fig. 2a) to record the electrical activity of the brain (EEG), and the Empatica E4 wristband (Fig. 2b) to record the electrodermal activity of the skin (EDA) and the blood volume pulse (BVP). The EEG device samples the signal at a frequency of 512Hz, the EDA at 4Hz, and the BVP at 64Hz.

The EEG sensor records the electrical activity of the brain through one electrode placed on the surface of the scalp. The cerebral waves can be categorized based on frequency as delta ($<$4Hz), theta (4-7,5Hz), alpha (4-12,5Hz), beta (13-30Hz), and gamma ($>$ 30Hz). Delta waves are mainly recorded during sleep, theta waves indicate a decrease of vigilance level, alpha waves are recorded during relaxing moments, beta waves are observed during mental activity demanding attention or concentration, and gamma waves are related to cognitive processes.

EDA is constituted by a tonic component, indicating the level of electrical conductivity of the skin (SCL), and a phasic component, representing the phasic changes in electrical conductivity or skin conductance response (SCR) [26].

BVP is the volume of blood that passes through tissues in a localized area with each heartbeat. It is used to calculate the heart rate (HR) and the variation in the time interval between heartbeats or heart rate variability (HRV).

Experimental protocol. We organized the experiment according to the following phases.

Pre-experimental briefing. The participant gets acquainted with the settings—e.g., sitting in a comfortable position, adjusting the monitor height. The experimenter summarizes the upcoming steps and explains how to perform the task. Subsequently, the participants signed the consent form for personal data treatment.

Device calibration. The participant wears the biometric sensors (see Figure 3) and watches a two-minute fish thank video to collect physiological baselines.[3]

Task execution. The participants performed 27 tasks, divided into three sessions. The tasks are displayed on a 24-inches monitor connected to a standard desktop computer. Answers are recorded by pressing the arrow keys (i.e., left arrow to accept and right arrow to reject). After each session, a 10-second fixation cross is displayed. Each session is composed of three unique code comprehension, and six unique prose comprehension tasks randomly displayed for 60 and 30 seconds respectively. The total duration of the experiment for one participant was 30 minutes.

Post-experimental briefing. The participants can ask questions and give feedback about the experiment. Finally, they were rewarded with a voucher for a meal.

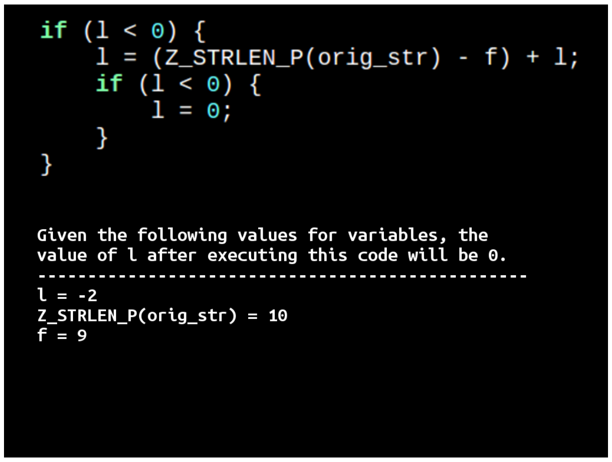
Machine Learning settings. We perform the classification using eight different machine learning algorithms whereas only one, Gaussian Process Classifier, was used in the original study. Moreover, we present two different validation settings, leave-one-run-out cross-validation (LORO-CV, as in the original study) and the additional Hold-out validation.

D. Experimental Sample

Our goal is to have a sample size comparable to the original study. Accordingly, we recruited 32 (28 males, four females) Bachelors' students from the Department of Computer Science. We applied a quota sampling strategy based on students expertise, measured through the number of credits obtained in courses where the C programming language (i.e., the language used for the code comprehension tasks) was used. At the time of the experiment, the average GPA (grade point average) of the students was 3.0 ($\pm$0.25).

Outliers and dropouts. Once the experiment was completed, but before analyzing the data, we discarded two participants because they failed to complete more than 30% of the tasks (e.g., they did not provide an answer within the time allotted). We interpreted this as a sign of inability or negligence in carrying out the experimental tasks. Moreover, due to a technical issue with the devices, we discarded two more participants. Therefore, we considered a total of 28 participants (24 males, four females) during data analysis.

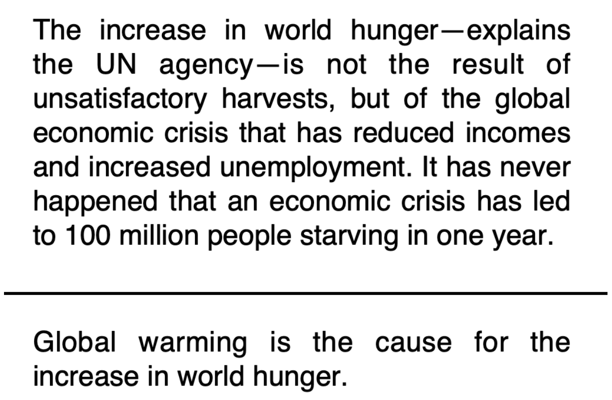
[3]Sensors 101 workshop. https://github.com/BioStack/Sensors101



```
if (l < 0) {
    l = (Z_STRLEN_P(orig_str) - f) + l;
    if (l < 0) {
        l = 0;
    }
}


Given the following values for variables, the
value of l after executing this code will be 0.
-----------------------------------------------
l = -2
Z_STRLEN_P(orig_str) = 10
f = 9
```

(a) Example of a code snippet used for the code comprehension task.



The increase in world hunger—explains the UN agency—is not the result of unsatisfactory harvests, but of the global economic crisis that has reduced incomes and increased unemployment. It has never happened that an economic crisis has led to 100 million people starving in one year.

Global warming is the cause for the increase in world hunger.

(b) Example of a prose snippet used for the prose comprehension task (translated from Italian to English).

Fig. 1: Examples of tasks for code and prose comprehension used in this study.

V. Machine Learning

In this section, we report the machine learning approach used to classify the comprehension tasks.

A. Dataset

Each of the 28 participants performed a total of 27 comprehension tasks (i.e., nine code, 18 prose). They had the possibility of not answering a task question if they did not feel confident. Therefore, out of the total 756 tasks, we consider the biometric signals associated with the 695 completed ones—469 of prose comprehension and 226 of code comprehension.

B. Preprocessing and Features extraction

The biometric signals were recorded during the entire experimental session for all the participants. However, to address our research questions, we only consider the signals associated with the stimuli of interest—i.e., the signals collected between the time a task appeared on the screen and the time the participant provided an answer. We did not consider signals collected when a participant was not focusing on a task—i.e., when a fixation point was displayed on the screen. To synchronize the measurement of the biometric signals with the tasks, we applied the following procedure.

(a) BrainLink EEG headset.

(b) Empatica E4 wristband.

Fig. 2: Devices used to measure biometric signals in this study.

Fig. 3: A participant wearing the wristband and headband during the experimental session.

1) We registered the timestamp at the start of the experiment (`t_start_experiment`);
2) We saved the name of the task, its type (code or prose), and the timestamp of the answer (`t_answer`);
3) We calculated the timestamp for the start of each task (`t_start`) using `t_start_experiment` and the duration associated with each type (60 seconds for code and 30 seconds for prose);
4) From each biometric signal, we selected the samples recorded between `t_start` and `t_answer`.

For each participant, we normalize the signals to her baseline using Z−score normalization [21]. As it is customary for this kind of studies [12], the baseline was calculated considering the last 30 seconds of the fish thank video showed to the participant during the device calibration.

Finally, to maximize the signal information and reduce noise caused by movement artifacts, we applied multiple filtering techniques, as reported in Figure 4.

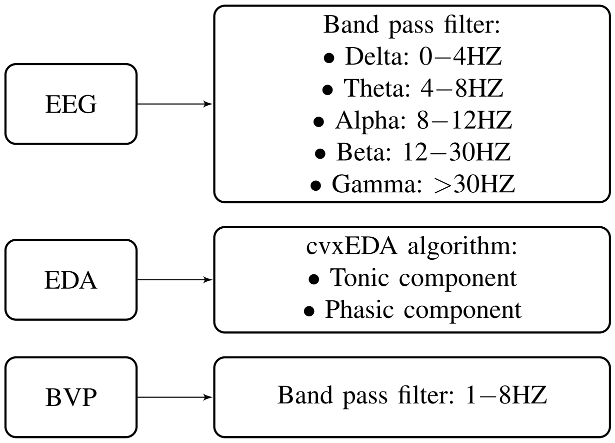


Fig. 4: Filtering strategies for the biometric signals (EEG, EDA, and BVP) collected in this study.

Regarding EEG and BVP, we extract the different frequency bands using a band−pass filter at different frequency intervals [27]. Concerning EDA, we applied the cvxEDA algorithm [28] to extract the tonic and the phasic components.

After signals preprocessing, we extracted the feature presented in Table II. We selected features based on previous studies [21] in which the same signals were used to train machine learning classifiers for recognizing affective and cognitive states of software developers.

In 21 cases, values were missing for HRV features due to noise in the recorded signals. As suggested in [29], missing values were replaced with the median of the non-missing values for that feature, calculated on the other tasks of the same type performed by the same participant.

C. Classification Settings

We choose eight popular machine learning classifiers (see Table III) based on previous studies using biometrical data [21], [30].

In the LORO setting (i.e., the same of the original study), the evaluation on a test set is repeated for 28 times—i.e., the number of participants in our dataset. At each iteration, we use all observations from *n-1* participants (i.e., 27) for training the model and test the performance on the remaining one.

In the Hold-out setting, we assess to what extent the trained model can generalize task classification on unseen new data from an hold-out test set not specific to a single participant. In such a setting, we split the entire dataset into training (20 participants) and test (8 participants) sets. The model is trained on the entire training set and then evaluated on the held-out test set. We repeat this process 10 times to further increase the validity of the results.

In line with previous research [31], [32], in both settings we performed hyper-parameter tuning using the `caret` R package[4]. Table III reports the parameters we tuned for to each classifier. For tuning, we followed a GridSearch approach [33] with `tuneLength` = 5—i.e., the maximum number of different values to be evaluated for each parameter [34], [35].

We evaluate the models using precision, recall, and F1-score for which we report values macro-averaged over the evaluation runs [36]. However, for comparison, we focus on the metric reported in the original study—i.e., balanced accuracy (BAC).

VI. Results

In this section, we present the empirical results obtained in our machine learning study (RQ_{Clf}, see Section VI-A) and investigate the relationship between expertise and task classification accuracy (RQ_{Exp}, see Section VI-B).

A. Classification of Comprehension Tasks based on Biometrics

We evaluate whether our models can classify which task a participant is performing based on signals combinations from different biometric sensors. In Figure 6a and Figure 6b, we compare the median BAC obtained on the different combination of signals by each classifier in the LORO and Hold-out settings respectively. A classifier which performs better than the others, independently from the signal(s) considered, cannot be observed. On the other hand, *nb* and *knn* classifiers do not seem appropriate for our classification task.

In Table IV and Table V, for each signal and their combination, we report the classifier with the highest BAC together with its precision, recall, and F-measure. In both evaluation settings, the EEG signal shows the worst performance (BAC = 66%, 67%). Conversely, Heart is the signal with the highest performance, yielding a BAC of 87% (90% in the Hold-out evaluation setting). Moreover, when combined with Hearth, also EEG and EDA performances increase considerably. This suggests that the best and most reliable classification results can be accomplished solely using the Empatica E4 sensors while the EEG contribution is negligible.

We answer RQ_{Clf} as follows.

Physiological signals can be used to train classifiers which accurately differentiate between code and prose comprehension tasks. The classifier trained using features based on Heart signal shows the best results.

B. Classification Accuracy and Expertise

We examine the relationship between classifier accuracy and participant expertise. For each participant, we consider the classifier with the best BAC among all configurations of classifiers and signals. We then calculate the association between BAC and the participant's GPA (for courses using C as reference language) using the Kendall tau correlation coefficient ($\tau \in [-1, -1]$ with 0 indicating no association).

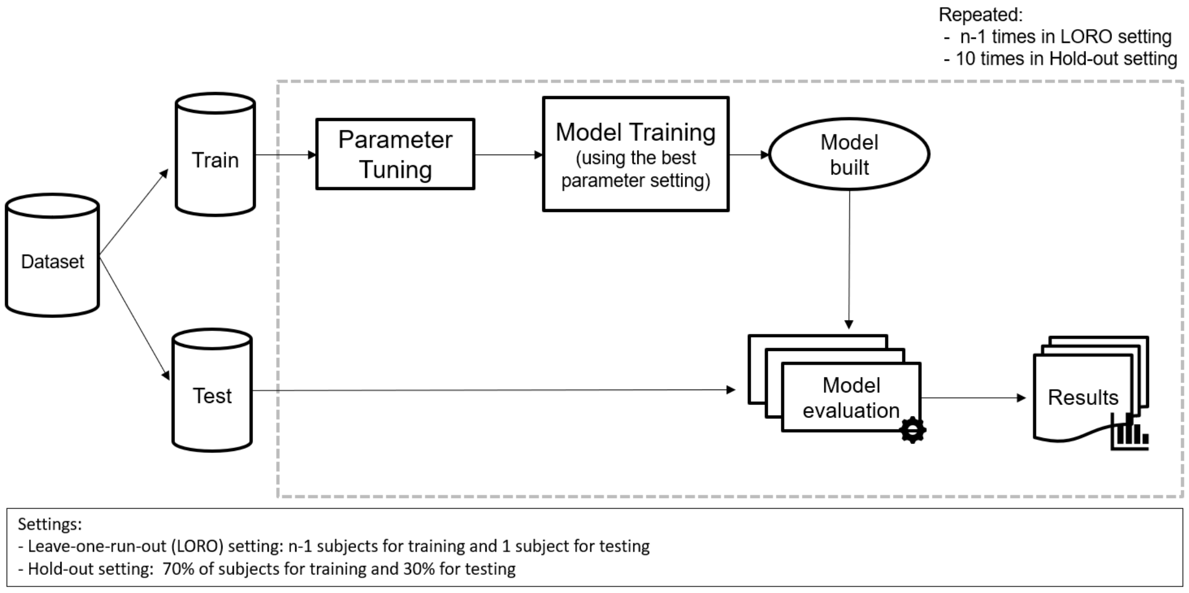
[4]http://topepo.github.io/caret/index.html

TABLE II: Machine learning features grouped by physiological signal.

	Signal	Features
Brain	EEG	● Frequency bin for alpha, beta, gamma, delta and theta waves ● Ratio between frequency bin of each band and one another ● For the attention and meditation measures: min, max, difference between the mean attention (meditation) during the baseline and during the task
Skin	EDA tonic EDA phasic	● mean tonic signal ● area under the curve (AUC) ● min, max, mean, sum peaks amplitudes
Heart	BVP HR HRV	● min, max, mean, sum peak amplitudes ● difference between the mean peak amplitude during baseline and during the task ● difference between the mean heart rate during the baseline and during the task ● difference between the variance heart rate during the baseline and during the task ● standard deviation of beat-to-beat (SDNN) intervals ● root mean square of the successive differences (RMSSD)

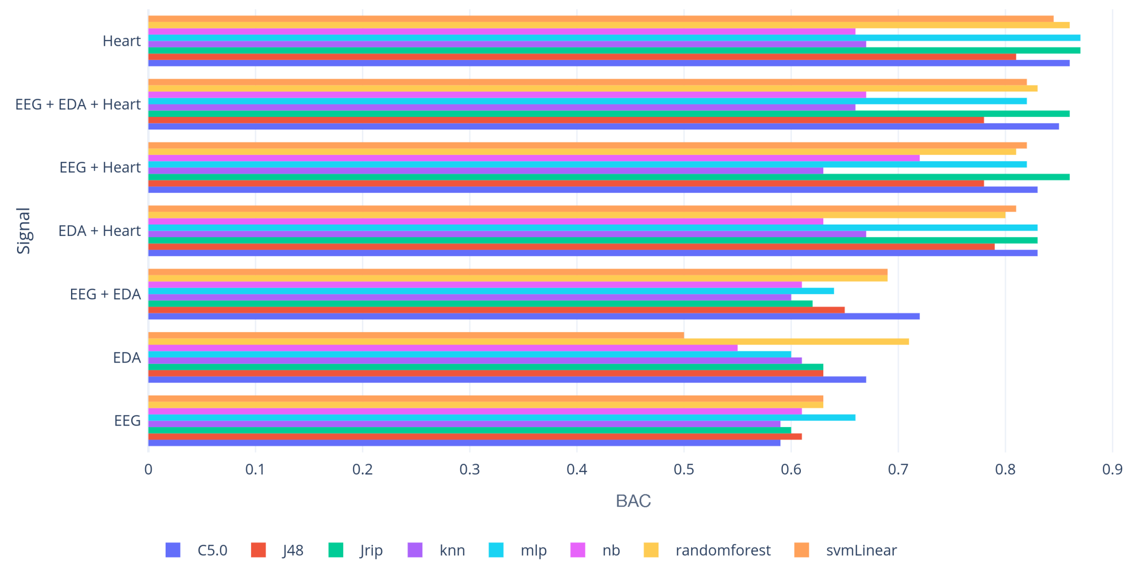
TABLE III: Machine learning classifiers used in this study and parameters used for tuning ("?" indicates a boolean parameter).

Family	Classifier (short name)	Parameter	Description
Bayesian	Naive Bayes (nb)	fL usekernel? adjust	Laplace correction factor Use kernel density estimate Bandwidth adjustment
Nearest Neighbor	K-Nearest Neighbor (knn)	k	#Clusters
Decision Trees	C4.5-like trees (J48)	C	Confidence factor for pruning
Support Vector Machines	SVM with Linear Kernel (svmLinear)	C	Cost penalty factor
Neural Networks	Multi-layer Perceptron (mlp)	size	#Hidden units
Rule−based	Repeated Incremental Pruning to Produce Error Reduction (Jrip)	NumOpt	#Optimizations iterations
Bagging	Random Forest (randomforest)	mtry	#Predictors sampled
Boosting	C5.0	trials model winnow?	# Boosting iterations Decision Trees or rule-based Apply predictor feature selection

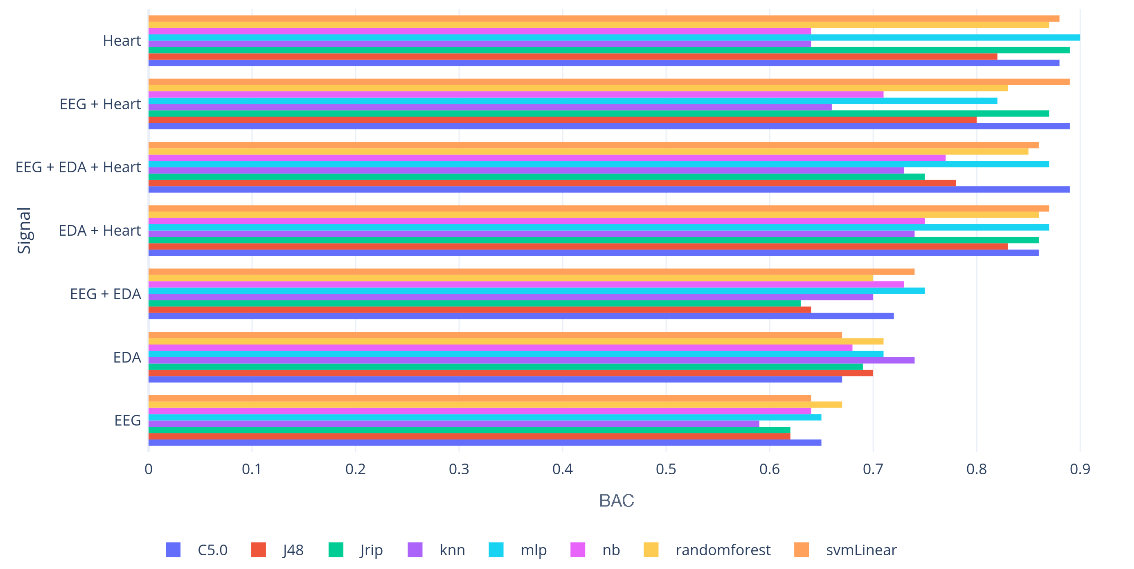

Repeated:
- n-1 times in LORO setting
- 10 times in Hold-out setting
Dataset
Train
Test
Parameter Tuning
Model Training (using the best parameter setting)
Model built
Model evaluation
Results

Settings:
- Leave-one-run-out (LORO) setting: n-1 subjects for training and 1 subject for testing
- Hold-out setting: 70% of subjects for training and 30% for testing

Fig. 5: Machine learning pipeline implemented in this study. The evaluation settings include LORO and Hold-out cross-validation.



(a) Median BAC of the machine learning classifiers after LORO cross-validation. Results are grouped according to different signal configurations.



(b) Median BAC of the machine learning classifiers after Hold-out cross-validation. Results are grouped according to different signal configurations.

Fig. 6: Median BAC of the machine learning classifiers evaluated in this study. Results are grouped according to different signal configurations.

For statistical testing ($\alpha = 0.05$), the null hypothesis assumes no association between the two variables.

There is a positive, although small, correlation between the classifier accuracy and the participants' expertise ($\tau = 0.16$), as shown in Fig. 7. However, we failed to reject the null hypothesis ($p = 0.31$). We answer RQ_{Exp} as follows.

Expertise is not related to the accuracy of classifiers trained using biometrical signals.

C. *Comparison with results of the original study*

RQ_{Clf} *replicates* the results of RQ1 from the original study. For comparison, we contrast our settings (code comprehension vs. prose comprehension) with the ones reported in the original study (code comprehension vs. prose review). We consider the best configuration of classifier and signal—i.e., `mlp` and Heart. Table VI reports the best classifier BAC for each class.

In both studies, prose-related tasks are better identified than code-related ones and with high accuracy—95% in the

TABLE IV: Results of the best machine learning classifier evaluated using LORO cross-validation.

Signal	**Best Classifier**	**Precision**	**Recall**	**F1**	**BAC**
EEG	mlp	0.72	0.66	0.62	0.66
EDA	rf	0.78	0.71	0.71	0.71
Heart	mlp	0.91	0.87	0.87	**0.87**
EEG + EDA	C5.0	0.75	0.72	0.72	0.72
EEG + Heart	Jrip	0.90	0.86	0.87	0.86
EDA + Heart	mlp	0.91	0.83	0.86	0.83
EEG + EDA + Heart	Jrip	0.88	0.86	0.86	0.86

TABLE V: Results of the best machine learning classifier evaluated using Hold-out cross-validation.

Signal	**Best Classifier**	**Precision**	**Recall**	**F1**	**BAC**
EEG	rf	0.70	0.67	0.68	0.67
EDA	Knn	0.83	0.74	0.77	0.74
Heart	mlp	0.95	0.90	0.92	**0.90**
EEG + EDA	mlp	0.75	0.75	0.75	0.75
EEG + Heart	C5.0	0.90	0.89	0.90	0.89
EDA + Heart	svm	0.93	0.87	0.89	0.87
EEG + EDA + Heart	C5.0	0.92	0.89	0.90	0.89

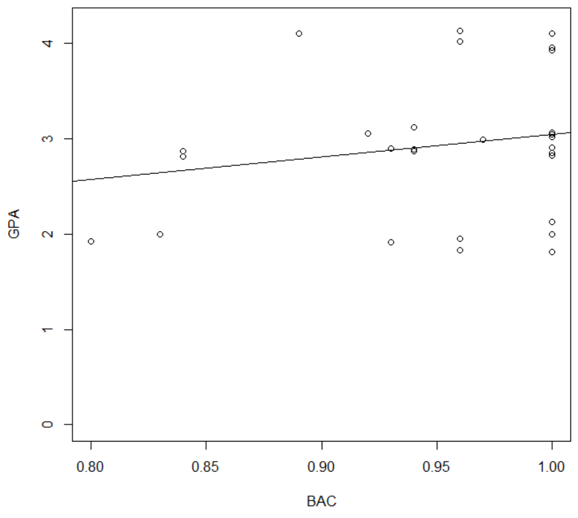


Fig. 7: Scatterplot of BAC vs. GPA for participants in this study. The regression line indicates a non-significant relationship.

original study and 99% in the replication. Our approach improves by 8% (9% if considering the Hold-out evaluation setting) the original study accuracy when classifying the same code comprehension tasks. Finally, the overall results show an improvement of 8% (11% if considering the Hold-out evaluation setting) obtained by using lightweight biometric sensors rather than fMRI.

We did *not replicate* RQ_{Exp}—i.e., RQ3 in the original study (see Table VI). The original study reported a negative correlation coefficient (Pearson r = -0.44, p = 0.016)

TABLE VI: Result comparison between the original study and this replication. For RQ_{Clf}, best BAC results for LORO (Hold-out) validation are reported.

RQ (original study)		**Original Study**	**This replication**	**Replicated?**
RQ_{Clf} (RQ1)	Overall	0.79	0.87 (0.90)	Yes, with improvements
	Code	0.72	0.80 (0.81)	
	Prose	0.95	0.99 (0.99)	
RQ_{Exp} (RQ3)		r = -0.44 (p = 0.01)	τ = 0.16 (p = 0.31)	No

between classifier accuracy and expertise. The inverse relationship between accuracy and expertise suggests that neural representations of code and prose are harder to differentiate once coding skills increase. Given the non-significant and small correlation coefficient reported in this study, a similar relationship between expertise and heart-related signals is not apparent. We believe that the reason for this result is the limited variation regarding experience within our sample. Compared to the original study sample (i.e., a mix of graduate and undergraduate students from different schools), our sample is more homogeneous (i.e., undergraduate students from a single school) which can be reflected in their expertise.

VII. Discussion

In this section, we discuss the limitations of our study, the current state of knowledge based on this and the original study results, and consider implications for practice and research.

A. Threats to Validity

Threats to the external validity of our replication are associated with the representativeness of the tasks. For code comprehension, we used the same tasks of the original study which, although targeting only the C language, were sampled from real-world projects. For prose comprehension, we use standardized text from the Italian Ministry of Education, Universities, and Research.

Our study suffers from threats to construct validity—i.e., how reliable are our measures in capturing comprehension for both code and prose. For the former, we use the same questions as in the original study [9], which are also utilized in previous work (e.g., [37]). For the latter, we use examples from cognitive linguistic studies [38], [39]. The complex biometric signals needed to be filtered before the analysis. To that end, we followed state-of-the-art practices from signal processing. The expertise construct is measured through a proxy, GPA, as in the original study. Although we acknowledge that software development expertise is difficult to measure, GPA correlates with learning and academic skills [40], and was measured taking into only courses focusing on the C language used during the experiment.

Nevertheless, when assessing the impact of expertise on classifiers performance, we did not observe a significant correlation. We cannot exclude that such result is due to the homogeneity of the sample (i.e., solely undergraduate students); thus, representing a potential threat to internal validity.

The validity of our conclusions is based on the reliability of the machine learning models and null-hypothesis statistical testing used to answer the research questions. Regarding machine learning, we mitigated conclusion validity threats by i) running several algorithms addressing the same classification task, ii) applying hyper-parameters tuning to optimally solve the task, and iii) reporting results from two different evaluation settings—i.e., LORO and Hold-out. Regarding hypothesis testing, we rely on robust, non-parametric statistics [41] for which the effect size (i.e., Kendall τ) can be interpreted similarly to the one reported in the original study (i.e., Pearson r).

B. Drawing Conclusion Across Studies

In this study, we strengthen the original study conclusion that different comprehension tasks can be recognized using biometric-based signals as a proxy for cognitive effort. In particular, we explicitly compare code comprehension and prose comprehension—strengthening the construct validity of the original study—while making our approach affordable in real-world settings by using lightweight sensors. Our setup costs less than \$2,000 as opposed to the \$21,000 reported in the original study. Considering performance on individual tasks, we show that prose comprehension can be more accurately (and in almost every case) recognized compared to code comprehension. Our best overall accuracy (90%) and the improvement over the original study (Δ = 8%, approx. \$2,400 saved for each percentage point gained in performance) is ground for further evaluation in in-vivo settings—e.g., integration with development environments.

Implications for Practice. Our results provide opportunities for improving software engineering tools. By means of a relatively cheap wristband (approx. \$1,500), the development environment can be made aware of the comprehension task in which a developer is engaged and optimize support for such a task. For example, knowing when developers are comprehending source code—a more demanding task than prose comprehension [12]—can be leveraged to measure their interruptibility better [11] and adjust their environment (e.g., by temporarily disabling notifications).

Using our approach, developers can collect interesting metrics from a Personal Software Process perspective [42], such as time spent comprehending source code vs. time spent understanding requirement specifications or documentation. Developers can leverage these metrics to improve their productivity, effort estimation, and planning skills.

Furthermore, a developer engaged for too long in comprehending a specification or a piece of code may indicate quality issues related to complexity [43] In turn, this approach can benefit from the integration with eye-tracking technology to identify the specific focus of a developer [44], [45] and recommend appropriate documentation to support her information needs [46], [10].

Implications for Research. Although previous studies have tackled the use of lightweight biometrics in software engineering, this study is one of the first to explicitly deal with code comprehension. Researchers can build on our result in several directions.

Currently, researchers study comprehension strategies (e.g., bottom-up vs. top-down) by relying on developers' assessment (e.g., subjective rating [47] or think aloud protocols [48]) or more invasive methods (e.g., eye-tracking [24] or fMRI [8]). The ability to automatically recognize code comprehension tasks using physiological signals enables less invasive research on comprehension strategies. Code comprehension is the basis for several other software development tasks [9]. Our approach can be used to study the "weight" that comprehension has for such tasks—e.g., refactoring [49] or code reviews [50]—in an unobtrusive (and cheaper) way.

Cognitive activities are related to task difficulty. As shown in the original study [9], understanding code is more difficult than comprehending text. Our study confirms previous work results which showed that it is possible to classify task difficulty using lightweight biometric sensors [12].

We showed that an EEG headset equipped with one electrode is not sufficient to recognize the task a participant is performing. Therefore, we suggest to researchers interested in the same goal of this study, but focusing on the investigation of neural activity measured unobtrusively through EEG, to invest in devices with higher definition (e.g., 14 or 32 channels).

Furthermore, researchers can replicate our setup using devices available at retail shops and standard data analysis tools.

VIII. Conclusion and Future Work

This paper presents the replication of a controlled experiment aimed at i) automatically classifying which kind of comprehension task (prose or code) a developer is performing and ii) studying the correlation between the classifier accuracy and developers' expertise.

In the original study, the authors explored the use of fMRI finding that it is possible to classify which task a participant is undertaking based on brain activity. However, collecting fMRI signals is expensive and can be applied only for *in-vitro* studies. Therefore, we investigated whether fMRI could be replaced by biometric signals, collected using lightweight devices, which previous research found to be correlated with cognitive effort. We found that an off-the-shelf EEG headset is not suitable to achieve our goal with high performance. Conversely, the heart activity, captured using a wristband, can be used to distinguish between code and prose comprehension tasks with high accuracy. The original study also showed that, when considering expert developers, the two tasks are harder to distinguish at neural level. We were not able to replicate this result using biometric signals in our homogeneous sample composed of undergraduate students. Further replications, involving a more heterogeneous sample, are required to further investigate the association between participants' expertise and the performance of the task classifiers.

Our future work consists in investigating software development expertise from a physiological perspective. Furthermore, we want to assess an additional task in which code and prose are mixed—i.e., technical documentation, programming tutorials, and StackOverflow posts.

Acknowledgment

This work is partially funded by the project "Emotion recognition based on biometrics" funded by the Department of Computer Science of the University of Bari.

References

[1] A. Von Mayrhauser and A. M. Vans, "Program Comprehension During Software Maintenance and Evolution," *Computer*, no. 8, pp. 44–55, 1995.

[2] D. Hendrix, J. Cross, and S. Maghsoodloo, "The Effectiveness of Control Structure Diagrams in Source Code Comprehension Activities," *IEEE Transactions on Software Engineering*, vol. 28, no. 5, pp. 463–477, 2002.

[3] M. Ciolkowski, O. Laitenberger, and S. Biffl, "Software Reviews: The State of the Practice," *IEEE software*, no. 6, pp. 46–51, 2003.

[4] F. Lanubile, T. Mallardo, F. Calefato, C. Denger, and M. Ciolkowski, "Assessing the Impact of Active Guidance for Defect Detection: A Replicated Experiment," in *null*. IEEE, 2004, pp. 269–279.

[5] D. Rombach, M. Ciolkowski, R. Jeffery, O. Laitenberger, F. McGarry, and F. Shull, "Impact of Research on Practice in the Field of Inspections, Reviews and Walkthroughs: Learning from Successful Industrial Uses," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 6, pp. 26–35, 2008.

[6] K. Nishizono, S. Morisakl, R. Vivanco, and K. Matsumoto, "Source Code Comprehension Strategies and Metrics to Predict Comprehension Effort in Software Maintenance and Evolution Tasks-An Empirical Study with Industry Practitioners," in *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. IEEE, 2011, pp. 473–481.

[7] A. Armaly, P. Rodeghero, and C. McMillan, "AudioHighlight: Code Skimming for Blind Programmers," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2018, pp. 206–216.

[8] 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 *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 378–389.

[9] B. Floyd, T. Santander, and W. Weimer, "Decoding the Representation of Code in the Brain: An fMRI Study of Code Review and Expertise," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 175–186.

[10] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Confusion in Code Reviews: Reasons, Impacts, and Coping Strategies," in *Proceedings of 26th International Conference on Software Analysis, Evolution and Reengineering SANER 2019*. IEEE Press, 2019, pp. 49–60.

[11] M. Züger, C. Corley, A. N. Meyer, B. Li, T. Fritz, D. Shepherd, V. Augustine, P. Francis, N. Kraft, and W. Snipes, "Reducing Interruptions at Work: A Large-scale Field Study of FlowLight," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 61–72.

[12] 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 *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 402–413.

[13] 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)*, 2018.

[14] D. W. Rowe, J. Sibert, and D. Irwin, "Heart Rate Variability: Indicator of User State as an Aid to Human-computer Interaction," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1998, pp. 480–487.

[15] M. Gjoreski, M. Luštrek, and V. Pejović, "My Watch Says I'm Busy: Inferring Cognitive Load with Low-Cost Wearables," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. ACM, 2018, pp. 1234–1240.

[16] J. C. Carver, "Towards Reporting Guidelines for Experimental Replications: A Proposal," in *1st International Workshop on Replication in Empirical Software Engineering Research*, 2010.

[17] M. T. Baldassarre, J. Carver, O. Dieste, and N. Juristo, "Replication Types: Towards a Shared Taxonomy," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014, p. 18.

[18] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The Role of Replications in Empirical Software Engineering," *Empirical software engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[19] N. Peitek, J. Siegmund, S. Apel, C. Kästner, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann, "A Look into Programmers' Heads," *IEEE Transactions on Software Engineering*, 2018.

[20] S. Radevski, H. Hata, and K. Matsumoto, "Real-time Monitoring of Neural State in Assessing and Improving Software Developers' Productivity," in *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 2015, pp. 93–96.

[21] S. Müller and T. Fritz, "Stuck and Frustrated or in Flow and Happy: Sensing Developers' Emotions and Progress," in *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, vol. 1. IEEE, 2015, pp. 688–699.

[22] C. Parnin, "Subvocalization-Toward Hearing the Inner Thoughts of Developers," in *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*. IEEE, 2011, pp. 197–200.

[23] S. Müller and T. Fritz, "Using (bio) Metrics to Predict Code Quality Online," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 452–463.

[24] Y. Ikutani and H. Uwano, "Brain Activity Measurement During Program Comprehension with NIRS," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*. IEEE, 2014, pp. 1–6.

[25] 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. Intl Symposium Empirical Software Engineering and Measurement (ESEM)*. ACM, 2018.

[26] J. J. Braithwaite, D. G. Watson, R. Jones, and M. Rowe, "A Guide for Analysing Electrodermal Activity (EDA) & Skin Conductance Responses (SCRs) for Psychological Experiments," *Psychophysiology*, vol. 49, no. 1, pp. 1017–1034, 2013.

[27] F. Canento, A.Fred, H. Silva, H. Gamboa, and A. Loureno, "Multimodal biosignal sensor data handling for emotion recognition," in *SENSORS, 2011 IEEE*, 2011, pp. 647–650.

[28] A. Greco, G. Valenza, A. Lanata, E. P. Scilingo, and L. Citi, "cvxEDA: A Convex Optimization Approach to Electrodermal Activity Processing," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, pp. 797–804, 2016.

[29] H. Trevor, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[30] S. Koelstra, C. Mühl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Yiannis) Patras, "Deap: A database for emotion analysis using physiological signals," *IEEE Transactions on Affective Computing*, vol. 3, pp. 18–31, 12 2011.

[31] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 321–332.

[32] ——, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering*, 2018.

[33] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-parameter Optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[34] M. Kuhn, "Building Predictive Models in R Using the caret Package," *Journal of Statistical Software, Articles*, vol. 28, no. 5, pp. 1–26, 2008.

[35] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An Empirical Comparison of Model Validation Techniques for Defect Prediction Models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.

[36] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002.

[37] J. Sillito, G. C. Murphy, and K. De Volder, "Questions Programmers ask During Software Evolution Tasks," in *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2006, pp. 23–34.

[38] J. Hatcher, M. J. Snowling, and Y. M. Griffiths, "Cognitive Assessment of Dyslexic Students in Higher Education," *British journal of educational psychology*, vol. 72, no. 1, pp. 119–133, 2002.

[39] E. Macaro, “Strategies for Language Learning and For Language Use: Revising the Theoretical Framework,” *The Modern Language Journal*, vol. 90, no. 3, pp. 320–337, 2006.

[40] W. A. Grove, T. Wasserman, and A. Grodner, “Choosing a Proxy for Academic Aptitude,” *The Journal of Economic Education*, vol. 37, no. 2, pp. 131–147, 2006.

[41] G. E. Noether, “Why Kendall Tau?” *Teaching Statistics*, vol. 3, no. 2, pp. 41–43, 1981.

[42] W. S. Humphrey, “Personal Software Process (PSP),” *Encyclopedia of Software Engineering*, 2002.

[43] X. Yang, R. G. Kula, N. Yoshida, and H. Iida, “Mining the Modern Code Review Repositories: A Dataset of People, Process and Product,” in *Proceedings of the 13th International Conference on Mining Software Repositories*. ACM, 2016, pp. 460–463.

[44] K. Kevic, B. Walters, T. Shaffer, B. Sharif, D. C. Shepherd, and T. Fritz, “Eye Gaze and Interaction Contexts for Change Tasks-Observations and Potential,” *Journal of Systems and Software*, vol. 128, pp. 252–266, 2017.

[45] B. Sharif, T. Shaffer, J. Wise, and J. I. Maletic, “Tracking Developers’ Eyes in the IDE,” *IEEE Software*, vol. 33, no. 3, pp. 105–108, 2016.

[46] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, and M. Linares-Vásquez, “On-demand Developer Documentation,” in *Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on*. IEEE, 2017, pp. 479–483.

[47] A. Dunsmore and M. Roper, “A Comparative Evaluation of Program Comprehension Measures,” *The Journal of Systems and Software*, vol. 52, no. 3, pp. 121–129, 2000.

[48] T. M. Shaft and I. Vessey, “The Relevance of Application Domain Knowledge: The Case of Computer Program Comprehension,” *Information systems research*, vol. 6, no. 3, pp. 286–299, 1995.

[49] J. Feigenspan, M. Schulze, M. Papendieck, C. Kästner, R. Dachselt, V. Köppen, M. Frisch, and G. Saake, “Supporting Program Comprehension in Large Preprocessor-based Software Product Lines,” *IET software*, vol. 6, no. 6, pp. 488–501, 2012.

[50] A. Van Deursen, “Program Comprehension Risks and Opportunities in Extreme Programming,” in *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*. IEEE, 2001, pp. 176–185.