Will you use software development support using biosignals? A survey from software developers

Ryo Soga*†, Hideyuki Kanuka*, Takatomi Kubo†, Takashi Ishio‡, Kenichi Matsumoto†
* *Research & Development Group, Hitachi, Ltd.*
{ryo.soga.jj, hideyuki.kanuka.dv}@hitachi.com
† *Graduate School of Science and Technology, Nara Institute of Science and Technology*
{takatomi-k, matumoto}@is.naist.jp
‡ *Department of Media Architecture, Future University Hakodate*
ishio@fun.ac.jp

***Abstract*—Biosignals reflect the mental states of software developers and could improve support technologies for software development activities. Although several technologies for software development support using biosignals (BioSDS) have been proposed, BioSDS has not yet been deployed in actual software development workplaces. As a prerequisite for industrial deployment, BioSDS must be well understood and accepted by software developers. However, the current level of their acceptance has not been comprehensively assessed. In this study, we conducted a survey to clarify the current level of acceptance of BioSDS and potential attributes that influence the level of acceptance. We defined eleven use-cases based on six previous primary studies related to BioSDS, and then asked developers at Hitachi, a Japanese IT company in the FORTUNE 500, about the level of acceptance of each use-case. Our analysis of eighty-six responses revealed that four out of eleven use-cases had some level of acceptance by software developers. In addition, we found four attributes that affect the level of acceptance: subject to be measured, objectives, interventions, and timing. These findings help to identify barriers to the adoption of BioSDS in the workplace.**

***Index Terms*—Biosignal, Acceptance, Software development support**

I. Introduction

Several technologies for software development support using biosignals (BioSDS) have been proposed but have not yet been deployed in workplaces of software development. For example, Züger et al. showed that software developers' interruptibility can be estimated from biosignals and proposed a use-case (UC) in which software developers can easily maintain a focused state by notifying their interruptibility to teammates [18]. Müller et al. also showed that biosignals can be used to estimate whether a developer's edits contain defects and proposed a UC that recommends peer review of codes containing defects [13]. These UCs can boost software development by considering the mental state of software developers [4].

As the prerequisites for the introduction of BioSDS to workplaces, software developers need to accept BioSDS. The acceptance to such new technologies has been described following the technology acceptance model (TAM) [2]. Following TAM, the acceptance of BioSDS could be evaluated from the agreement level with the the self-predicted future usage.

In this study, we conducted a survey of software developers at a large Japanese IT company to determine the current level of acceptance of BioSDS and to explore attributes that influence the acceptance. We analyzed BioSDS UCs presented by six previous primary studies and extracted four candidate attributes of UCs affecting the acceptance: (i) subject to be measured, (ii) objective, (iii) intervention and (iv) timing. We defined 11 BioSDS UCs varying attributes (ii)–(iv) , and designed questionnaires that asks the acceptance level of each UC. In addition, for (i), two types of questionnaires were created: whether the subject was the respondent or not.

We obtained 86 responses and analyzed them to answer the following two research questions (RQs).

RQ1. Do software developers accept BioSDS?
RQ2. How do attributes of UCs affect the level of acceptance?

These findings contribute to reveal the obstacles to deploy BioSDS in workplaces.

II. Related Work

A. Review of software engineering studies using biosignals

Several review studies were presented previously [6], [10], [15]. Weber et al. identified 89 studies in the field of software engineering that used neural activity, such as brain activity and autonomic nervous systems, and showed that there are 4 types of contributions that can be made by the study of neural activity, [15]: (1) contributions to understanding human factors (2) improvement in software development, (3) investigation of software-understanding methods, and (4) development of software systems that adapt to the user's mental state. Menzen et al. identified 40 major studies that used biosignals in software engineering, categorized their types and themes, and pointed out the lack of application in real-world development environments [10]. Gonçales et al. selected 33 studies that used

DOI reference number: 10.18293/SEKE2023-050

biosignals to estimate cognitive load and investigated the challenges in realistic scenarios, noting the lack of accuracy in machine learning, [6]. These studies are similar to ours in that they compared studies that used biosignals in software engineering and attempted to identify challenges in existing research. However, they did not consider whether they are accepted by software developers.

B. *Evaluating software-development support technologies*

Several studies proposed new support technologies and evaluated usefulness of them [1], [5], [17]. In a study that did not use biosignals, Züger et al. asked 449 knowledge workers from 12 countries to use a light to make it known they did not want to be disturbed [17]. During the first half of the five-week experiment, they conducted a survey asking whether the participants wanted to stop using the light and indicated their willingness to continue after they were allowed to use the light. They were also surveyed about whether they wanted to continue using it. There are two studies that used eye gaze as a biosignal. Glücker et al. implemented EyeDE, which uses eye trackers to browse related parts of a program displayed in an integrated development environment (IDE) with the gaze. Four people were interviewed about their qualitative impressions of EyeDE and found it interesting [5]. Ahrens et al. also developed a tool to assist novice users in understanding a program by displaying a heatmap on the IDE that shows the viewing positions of the eyes viewed by expert users. Feedback on this tool was diverse, including responses that it reduced comprehension difficulty and that it was intrusive, [1]. However, these were only usefulness evaluations for individual UCs. Quantitative evaluations when comparing UCs have not been conducted.

III. Method

The acceptance of BioSDS was surveyed by defining comprehensive BioSDS UCs and presenting them to software developers. Four candidate attributes of BioSDS UCs were extracted from 6 previous primary studies: (i) subject to be measured, (ii) objective, (iii) intervention and (iv) timing. Comprehensive BioSDS UCs were defined as varying 3 attributes (ii)–(iv). Subsequently, questionnaires were designed to ask the acceptance of each UC and the reasons of selecting the most or the least useful UC. Besides, for (i), the questionnaires were separately asked whether the subject was the respondent or not. Eventually, responses were analyzed to answer 3 RQs.

A. *Attributes of BioSDS UCs*

1) Select primary BioSDS studies: Six primary BioSDS studies were selected (Table I) from studies cited by the most cited review study [15] among previous review studies of software engineering studies using biosignals [6], [10], [15] referred in II-A.

The most cited review study constructed the search query by combining 22 words related to neural activity and 13 words related to software engineering, and obtained 89 studies. Then, the review study classified obtained 89 studies into 5 categories: empirical studies (N=47), empirical (research in progress) (N=24), methodological studies (N=8), conceptual studies (N=5), and review studies (N=5). Empirical studies were further categorized into studies in which biosignals were explanatory variables (N=19) and those in which biosignals were the target variable (N=26).

We selected 6 primary studies satisfying two criteria from 19 studies in which biosignals were explanatory variables. Firstly, 10 studies were excluded because the number of citations was less than 10. Secondly, 3 studies were excluded because they did not explicitly present UCs. Consequently, 6 studies were remained and their number of citations were 25–158(Table I). The number of citations was retrieved from Scopus (https://www.scopus.com) in May 2022.

2) Extract UCs and their attributes from six primary BioSDS studies: Fifteen UCs were extracted from six primary BioSDS studies (Table I). All UCs analyzed biosignals to detect the stress or the cognitive load of software developers, and attempted to boost software development by interventions. From 12 UCs, we extracted four candidate attributes: (i) subject to be measured, (ii) objective, (iii) intervention and (iv) timing.

(i) All UCs had two types of subjects: subjects to be measured and subjects not to be measured and two types of subjects would have totally different experiences in UCs. For example, “Assistance in preventing interruption” had at least two subjects and one subject was measured and prevented from the other’s interruptions. Therefore, the acceptance would be varying by whether “subject to be measured = respondent” or “subject to be measured = non-respondent”.

(ii) The UCs’ objectives were categorized into following eight types. “O1. Assessing quality”, which had 2 UCs, estimated the quality of programs and attempted to identify codes to be reviewed. “O2. Assessing skills”, which had 1 UC, estimated the skills of software developers. “O3. Preventing bugs”, which had 5 UCs, estimated the quality of programs and attempted to prevent bugs by double-checking the program. “O4. Preventing interruptions”, which had 2 UCs, estimated the interruptibility of software developers and attempted to prevent interruptions from other software developers. “O5. Searching code”, which had 1 UC, estimated the cognitive load of software developers and attempted to present programs related to codes causing the high cognitive load. “O6. Taking breaks”, which had 2 UCs, estimated the stress of software developers and attempted to encourage software developers to take a break. The acceptance would be varying by whether each objective satisfied needs of each software developer.

(iii) The UC’s interventions were categorized into 3 types, and each type of interventions had different impact

TABLE I
BioSDS UCs proposed in six primary studies

Ref.	UC	Objective	Intervention	Timing
[3]	Prevent commits with bugs	O3. Preventing bugs	–	Non-real-time
[12]	Identify code to be reviewed	O1. Assessing quality	I3. Assessment	–
	Search related code	O5. Searching code	I1. Notification (Private)	Real-time
	Assist to take a break	O6. Taking breaks	I1. Notification (Private)	Real-time
	Prevent interruption	O4. Preventing interruptions	I2. Notification (Team)	Real-time
[18]	Notify when not to interrupt	O4. Preventing interruptions	I2. Notification (Team)	Real-time
[13]	Prevent commits with bugs	O3. Preventing bugs	–	Non-real-time
	Identify code to be reviewed	O3. Preventing bugs	I2. Notofication (Team)	Non-real-time
	Assist to take a break	O6. Taking breaks	I1. Notification (Private)	Real-time
[7]	Prevent commits with bugs	O3. Preventing bugs	–	Non-real-time
	Annotate difficult code	O1. Assessing quality	I3. Assessment	–
[9]	Prevent bugs due to difficult tasks	O3. Preventing bugs	–	Real-time
	Estimate programming experience	O2. Assessing skills	I3. Assessment	–

on software development. Both "I1. Notification (Private)" and "I2. Notification (Team)" intervened software development by notifying estimation results to individuals or teammates, and their impacts would be relatively low because software developers could freely ignore notifications. "I3. Evaluation" intervened software development by evaluating skills or the quality of programs and its impact would be relatively high because additional works would be required when the evaluation results was bad. The acceptance would be varying by whether software developers tolerated those impacts.

(iv) The UC's timings of intervention were categorized into 2 types: "Real-time" and "Non-real-time", and two types of timings varied frequencies and adequacy. Interventions in real-time could help software developers immediately when they had troubles, but too much interventions could ironically cause additional stress or interrupt self-help efforts. Therefore, the acceptance would be varying depending on the intervention timing.

B. Comprehensive BioSDS UCs

Based on extracted 12 BioSDS UCs and 4 attributes, 11 UCs were defined (Table II). Six UCs were defined to consider the most common objective "O3. Preventing bugs" (Table II UC03, UC04, UC06, UC07, UC10, UC11) and 5 UCs were defined to consider other 5 different objectives(Table II UC01, UC02, UC05, UC08, UC09).

Six UCs related to "O3. Preventing bugs" were defined as varying (iii) interventions and (iv) timing. Four UCs were defined by combining two types of interventions: "I1. Notification (Private)" and "I2. Notification (Team)", and two types of intervention timing: "Real-time" and "Non-real-time" (Table II UC03, UC04, UC06, UC07). Besides, to further investigate how interventions affect the acceptance, we added the intervention category "I4. Edit" and defined 2 original UC scenarios (Table II UC10, UC11).

C. Questionnaires

We created two descriptions for each UC with different subjects to be measured: respondent and non-respondent. For the UCs whose subject is respondent, we use the descriptions in Table II. For the UCs whose subject is non-respondent, we replaced "(you)" with "(someone other than you)" in the descriptions. Each description is provided with a Likert question that asks the acceptance level of the description. We asked the same questions separately for those two groups of descriptions to understand the impact of the subject to be measured.

A Likert question was designed to evaluate "self-predicted future usage" as the acceptance. The actual question was "If the technology were available, would you use it in your future tasks?" and had 5 options: (1) "strongly disagree", (2) "disagree", (3) "neutral", (4) "agree", (5) "strongly agree". Likert responses higher than 3 suggest acceptance and that lower than 3 suggest rejection.

Questionnaires consisted of three pages. The first page provided the summary and prerequisites of survey. As the prerequisites for survey, "Do not care the feasibility" was written because software for realizing any of defined UCs is not yet available although software for collecting biosignals is available [11], [14]. The second page asked the Likert questions for descriptions whose subject to be measured is respondent. The third page asked the same questions for descriptions whose subject to be measured is non-respondent. All sentences in questionnaires were written in Japanese.

D. Analysis

1) RQ1. Do software developers accept BioSDS?: If software developers accept a UC of BioSDS, their Likert responses for a UC scenarios should be greater than 3 (neutral). We conducted Wilcoxon signed rank test [16] to analyze whether the median of Likert responses was significantly greater than 3 or not.

TABLE II
UCs created in this study

#	Description of UCs	Reference	Objective	Intervention	Timing
UC01	Prompt the developer (you) to take a break when the developer (you) feels stressed.	[12]	O6. Taking breaks	I1. Notification (Private)	Real-time
UC02	Present a candidate related code to the developer (you) only when the developer (you) feels stressed because it is difficult to find a related code.	[12]	O5. Searching code	I1. Notification (Private)	Real-time
UC03	While the developer (you) is viewing or modifying a code, identify the code that caused the stress in real-time and encourage the developer (you) to review the modification or consult with others.	[3], [7], [9], [13]	O3. Preventing bugs	I1. Notification (Private)	Real-time
UC04	Prompt the developer (you) to review the changes or consult with others before reflecting the changes in the production code after the developer (you) has completed modifying the code.	[3], [7], [9], [13]	O3. Preventing bugs	I1. Notification (Private)	Non-real-time
UC05	Let teammates know in real-time when the developer's (your) concentration level is high to prevent interruptions from those around the developer (you).	[12], [18]	O4. Preventing interruptions	I2. Notification (Team)	Real-time
UC06	While the developer (you) is modifying the code, identify the code that caused the stress in real-time and inform co-editors or teammates of the high stress of the developer (you).	[12], [13]	O3. Preventing bugs	I2. Notification (Team)	Real-time
UC07	Prompt the reviewer to conduct a focused code review of the code that caused the developer (you) stress after the developer (you) has completed the task but before it is reflected in the production code.	[12], [13]	O3. Preventing bugs	I2. Notification (Team)	Non-real-time
UC08	Estimate the quality of the refactoring based on the stress while the developer (you) is viewing the code to evaluate how the readability is improved from the developer's (your) perspective.	[7], [12]	O1. Assessing quality	I3. Assessment	–
UC09	Estimate technical skills on the basis of the stress while the developer (you) is viewing and modifying the code.	[9]	O2. Assessing skills	I3. Assessment	–
UC10	Edit code conventions to prevent program patterns that caused the developer (you) stress by identifying the code that caused the stress in real-time while the developer (you) is viewing the code.	–	O3. Preventing bugs	I4. Edit	Real-time
UC11	Submit issues for refactoring the code that caused the developer (you) stress by identifying that code in real-time while the developer (you) is viewing the code.	–	O3. Preventing bugs	I4. Edit	Real-Time

2) RQ2. How do attributes of UCs affect the acceptance?: The difference in Likert responses was analyzed depending on the four attributes of UCs: (i) subject to be measured, (ii) objectives, (iii) interventions, and (iv) timing. For (i) the subject to be measured, the responses were divided into two groups, and Wilcoxon signed rank test was conducted as a test for two related paired samples. For (ii) objectives and (iii) interventions, the response values were divided into multiple groups, and the Kruskal-Wallis test [8] was conducted as a multiple comparison test. When the test result was significant, Wilcoxon rank-sum test was conducted for every two groups. For (iv) timing, to conduct a test for two related paired samples, we limited the objective to "O3. Preventing bugs" and the intervention of "Notification (I1, I2)", and Wilcoxon signed rank test was conducted.

E. Distribution

The survey was distributed to employees at Hitachi, Ltd., a large Japanese IT company in FORTUNE 500; the company has more than 3,000 employees and has been doing business for more than 100 years in the electric device field. The survey was distributed to about 100 software developers in the financial industry and about 60 employees in the research department.

IV. Results

We obtained 86 responses, of which 52 were from the development sector and 34 from the R&D sector. The average response time was 38 minutes. The respondents' years of work experience were widely distributed from 0 to 35 years, with a particularly large number of respondents around 0 and 20 years (Fig. 1A). This indicates that this survey could reflect the opinions from both novice and experienced software developers.

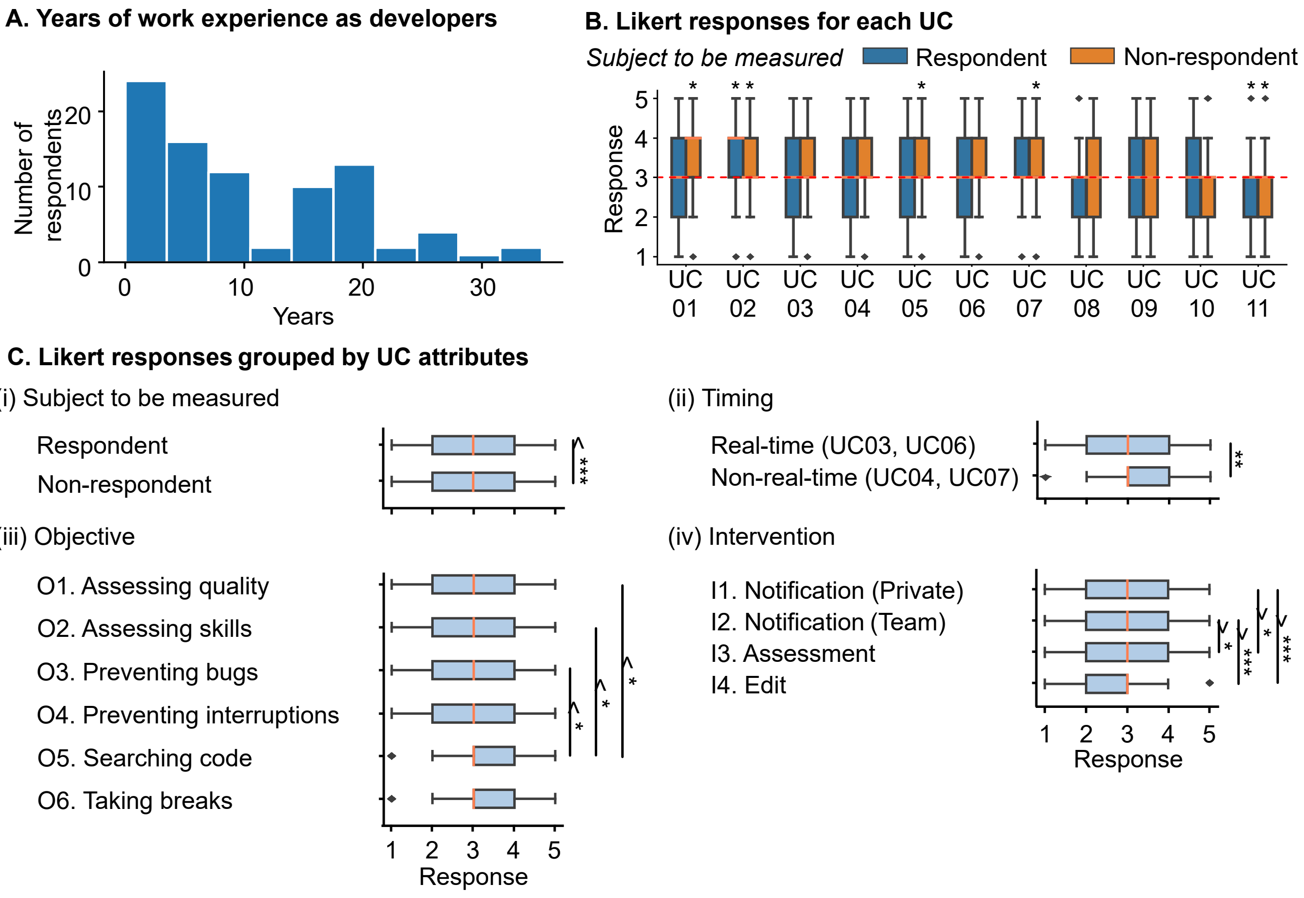

Fig. 1. Results of the survey*: p < 0.05, **: p < 0.001, ***: p < 0.0001, n.s.: not significant

A. RQ1. Do software developers accept BioSDS?

We verified whether the median Likert response of each of the 11 UCs was greater than 3 when the subject to be measured was either a respondent or non-respondent (Fig. 1B). As the result, the median was significantly greater than 3 for 4 use-cases (UC01, UC02, UC05, and UC07), and for 1 use-case (UC11) the median was significantly less than 3. This result indicates that developers accept some BioSDS UCs.

B. RQ2. How do attributes of UCs affect the level of acceptance?

For each of the 4 attributes of UC defined in this study, we grouped Likert responses by attribute values and compared median values of each group (Fig. 1C).

1) Subject to be measured: The median Likert response was significantly higher (Fig. 1C(i), $p = 5.34 \times 10^{-5} < 0.0001$) when the subject to be measured was non-respondent than when the subject was respondent. This indicates that the acceptance is higher when the software developer is not measured.

2) Timing: The median Likert response was significantly higher (Fig. 1C(ii), $p = 0.00132 < 0.05$) when the intervention timing was non-real-time than when the intervention timing was real-time. This indicates that developers could be more accepting of BioSDS with "Notification" interventions (I1, I2) and "O3. Preventing bugs" objective.

3) Objective: The median Likert response showed a significant difference among 6 types of objectives (Fig. 1C(iii), $p = 2.83 \times 10^{-5} < 0.0001$). Among fifteen pairs of two of the six objectives, 3 were significantly different after Bonferroni correction: $O1 < O5$ ($p = 0.00571 < 0.05$), $O2 < O5$ ($p = 0.00622 < 0.05$), and $O3 < O5$ ($p = 0.00142 < 0.05$). This indicates that "O5. Searching code" is the most acceptable, and "O4. Preventing interruptions" and "O6. Taking breaks" are moderately acceptable.

4) Intervention: The median Likert response showed a significant difference among 4 types of interventions (Fig. 1C(iv), $p = 1.19 \times 10^{-12} < 0.0001$) as well as the objectives. Among six combinations of two of the four interventions, four combinations were significantly different after Bonferroni correction; $I1 > I3 (p = 0.00148 < 0.05)$, $I1 > I4$ ($p = 3.11 \times 10^{-10} < 0.0001$), $I2 > I3$ ($p = 0.00627 < 0.05$), and $I2 > I4$ ($p = 1.26 \times 10^{-8} < 0.05$). This indicates that the two "Notification" interventions (I1, I2) are more acceptable than the other interventions (I3, I4).

V. Discussion

We examined the level of acceptance for each of the UCs based on previous BioSDS studies, and revealed the current level of acceptance (RQ1) and how four attributes affect the level of acceptance (RQ2).

First, developers accepted some BioSDS UCs to a certain extent (Fig. 1B). This result suggests that at least some BioSDS UCs are ready to be deployed in the workplace from an acceptance perspective. Currently, even if developers want to start using BioSDS, they cannot try out BioSDS UCs because it is difficult to prepare machine learning models trained by BioSDS studies. In addition, some measurement devices, such as eye trackers, are too expensive for all software developers to install. In the future, it is expected that at least some of the use-cases can be deployed in the workplace by publishing research-trained models or by providing inexpensive devices.

Second, all 4 of the attributes defined in this study affect the level of acceptance (Fig. 1C). For each of these attributes, respondents prefer values where the impact is relatively small. For example, acceptance is higher when the software developer is not measured (Fig. 1C(i)). In the future, we should clarify the reason by conducting a qualitative analysis.

VI. Conclusion

The acceptance of software development support technologies using biosignals (BioSDS) was investigated by presenting eleven defined use-cases to software developers in a large Japanese IT company. As a result of the analysis of 86 responses, 2 findings were obtained.

1) Four out of 11 BioSDS use-cases (UC01, UC02, UC05, UC07) were accepted by the software developers.
2) The level of acceptance of BioSDS use-cases varied depending on four attributes: the subject to be measured, the objectives, the interventions, and the timing. For all these attributes, software developers preferred low-impact values for them.

These results suggest that at least some use-cases are ready to be deployed in the workplace from an acceptance perspective, and deployment could be advanced by publishing assets created in BioSDS studies. On the other hand, the acceptance of BioSDS use-cases with high impact was still low, and we should clarify the reason by conducting qualitative analysis in the future.

As a limitation of this study, it is not clear how acceptance varies depending on the characteristics of software developers, as the respondents were a heterogeneous group. In the future, we will use a mathematical method for clustering heterogeneous groups to reveal which type of software developers tend to be more accepting of the use of BioSDS.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Numbers JP20H05706.

References

[1] M. Ahrens, K. Schneider, and M. Busch, "Attention in software maintenance: An eye tracking study," in *IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*, 2019, pp. 2–9.

[2] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989.

[3] 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*, 2014, pp. 402–413.

[4] T. Fritz and S. C. Müller, "Leveraging biometric data to boost software developer productivity," in *IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*, 2016, pp. 66–77.

[5] H. Glücker, F. Raab, F. Echtler, and C. Wolff, "Eyede: gaze-enhanced software development environments," in *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, 2014, pp. 1555–1560.

[6] L. Gonçales, K. Farias, B. d. Silva, and J. Fessler, "Measuring the cognitive load of software developers: A systematic mapping study," in *IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 2019, pp. 42–52.

[7] M. V. Kosti, K. Georgiadis, D. A. Adamos, N. Laskaris, D. Spinellis, and L. Angelis, "Towards an affordable brain computer interface for the assessment of programmers' mental workload," *International Journal of Human-Computer Studies*, vol. 115, pp. 52–66, 2018.

[8] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[9] S. Lee, D. Hooshyar, H. Ji, K. Nam, and H. Lim, "Mining biometric data to predict programmer expertise and task difficulty," *Cluster Computing*, vol. 21, no. 1, pp. 1097–1107, 2018.

[10] J. P. Menzen, K. Farias, and V. Bischoff, "Using biometric data in software engineering: a systematic mapping study," *Behaviour & Information Technology*, vol. 40, no. 9, pp. 880–902, 2021.

[11] A. N. Meyer, G. C. Murphy, T. Zimmermann, and T. Fritz, "Design recommendations for self-monitoring in the workplace," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, pp. 1–24, 2017.

[12] S. C. Müller and T. Fritz, "Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress," in *IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 688–699.

[13] ——, "Using (bio)metrics to predict code quality online," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 452–463.

[14] T. R. Shaffer, J. L. Wise, B. M. Walters, S. C. Müller, M. Falcone, and B. Sharif, "iTrace: enabling eye tracking on software artifacts within the IDE to support software engineering tasks," in *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 954–957.

[15] B. Weber, T. Fischer, and R. Riedl, "Brain and autonomic nervous system activity measurement in software engineering: A systematic literature review," *Journal of Systems and Software*, vol. 178, p. 110946, 2021.

[16] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[17] 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 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 61–72.

[18] M. Züger and T. Fritz, "Interruptibility of software developers and its prediction using psycho-physiological sensors," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 2981–2990.