Software code complexity assessment using EEG features

J. Medeiros[1], R. Couceiro[1], J. Castelhano[2], M. Castelo Branco[2], G. Duarte[1], C. Duarte[2], J. Durães[1,3], H. Madeira[1], P. Carvalho[1], and C. Teixeira[1]

Abstract— **This paper provides a study using Electroencephalography (EEG) to investigate the brain activity during code comprehension tasks. Three different code complexity levels according to five complexity metrics were considered. The use of EEG for this purpose is relevant, since the existing studies were mostly focused on neuroimaging techniques. Using Leave-One-Subject-Out cross-validation procedure for 30 subjects, it was found that the features related with the Gamma activity were the most common in all the folds. Regarding the brain regions, right parietal was the most frequent region contributing with more features. A Linear Discriminant Analysis Classifier for task classification, obtained a F-Measure of 92.71% for Code complexity easy, 52.25% for Code complexity intermediate and 53.13% for Code complexity advanced, revealing an evidence of mental effort saturation with the code complexity degree. This suggests that current code complexity metrics do not capture cognitive load and might not be the best approach to assess bug risk.**

I. INTRODUCTION

The stage of code comprehension is of great importance in software engineering and in the daily work of a programmer, either for fixing bugs, reusing or changing code. Any of these tasks requires human ability to comprehend different levels of knowledge such as logical thinking, mathematical, symbol manipulation and language skills [1]. Studies revealed that programmers spend around 70% of their time on code comprehension [2]. Furthermore, code quality is intimately connected to the code complexity and to the ability of programmers to understand, and master, such complexity. For this reasons it is essential to understand the relation between the neural processes involved in code comprehension and the metrics used for assessing code complexity.

There are many code complexity metric methods for assessing codes complexity and comprehensibility, namely MaCabe [3] and Halstead [4] metrics. These metrics are used worldwide for many purposes, mainly in the context of software testing but also including uses such as identifying parts of software that might need re-writing, improving code quality and robustness, reducing and estimating maintenance costs, predicting the error rate, among others [5].

[1]J. Medeiros, R. Couceiro, G. Duarte, J. Durães, H. Madeira, P. Carvalho and C. Teixeira are with with the Centre for Informatics and Systems of the University of Coimbra, Polo II, University of Coimbra, 3030-290, Coimbra, Portugal, (e-mails: juliomedeiros@dei.student.uc.pt, duarte.1995@live.com.pt, jduraes@isec.pt, {rcouceir, henrique, carvalho, cteixei}@dei.uc.pt).

[2]J. Castelhano, M. Castelo Branco and C. Duarte are with Institute of Nuclear Sciences Applied to Health, University of Coimbra, 3000-548, Coimbra, Portugal, (e-mails: joaocastelhano@uc.pt, mcbranco@fmed.uc.pt, catarinaduarte86@gmail.com).

[3]J. Durães is with Polytechnic Institute of Coimbra, 3045-093, Coimbra, Portugal, (e-mail: jduraes@isec.pt).

Electroencephalography (EEG) can be a powerful tool in this study, since it records electrical activity of the brain, which is one of the most relevant physiological areas analyzed when assessing cognitive workload [6]. Furthermore, other advantages of this monitoring method include its noninvasive nature, high time resolution, low cost and portability [7]. Therefore, in this study EEG recordings are used to understand the neural mechanisms involved in code comprehension of different complexity levels.

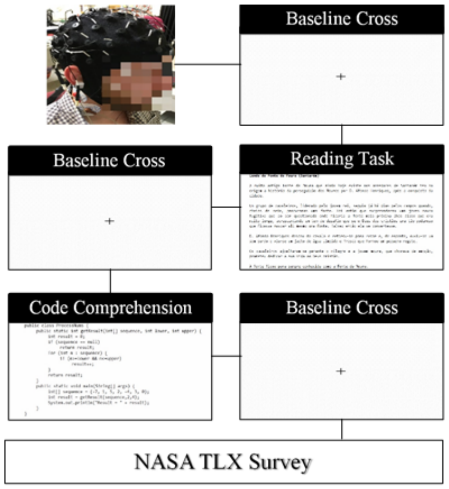
There are already some studies concerning cognitive load during code comprehension tasks and the brain regions activation, but most of studies were based on using neuroimaging techniques as fMRI or fNIRS [1], [8], [9]. However, this techniques can not allow direct measurements in a daily routine like the EEG can offer. In this area, there are few recent studies using EEG for assessing the mental workload of programmers during understanding and inspection of program codes for syntax errors [10]. Other recent researches focused on analyzing the difference on expertise of the programmers during code comprehension through EEG signals [11], [12]. However, these studies presented results derived from a small number of participants, and moreover, none of them compared the results of the mental workload with the code complexity metric methods.

II. DATA

EEG were recorded from 30 subjects. The subjects were submitted to three different trials of code comprehension tasks using three code snippets in JAVA, having each code snippet different complexity levels. Each trial consisted of a baseline task of text reading (in natural language) and a task of code comprehension of a certain complexity level. Before and after each task an empty screen with a cross in the middle was shown to the subject as a control interval for the next task. After each trial, the subject filled a survey designed using NASA TLX [13] to assess the subjective mental load perceived by the subject while doing the code comprehension task. This acquisition procedure is represented in Fig. 1. Regarding the order of the trials, it was according to an increasing degree of complexity. Fig. 2, inserted in section of Results and Discussion, shows the five software complexity metrics of the three code snippets.

All the participants signed a written consent for the data collection, which received clearance by the Ethics Committee of the Faculty of Medicine of the University of Coimbra, in accordance with the Declaration of Helsinki.

EEG was recorded, using the Neuroscan cap device with a sampling frequency of 1000Hz and 64 channels placed



```
public class ProcessNums {
    public static int getResult(int[] sequence, int lower, int upper) {
        int result = 0;
        if (sequence == null)
            return result;
        for (int n : sequence) {
            if (n>=lower && n<=upper)
                result++;
        }
        return result;
    }
    public static void main(String[] args) {
        int[] sequence = {-7, 1, 5, 2, -4, 3, 0};
        int result = getResult(sequence,2,4);
        System.out.println("Result = " + result);
    }
}
```

Fig. 1: Diagram of one trial procedure with an empty screen with a fixed cross and a reading task as baseline for analysis and a code comprehension task. The procedure was repeated for the three different trials.

according to the 10-20 system, including the reference channels M1, M2, CB1 and CB2. The Neuroscan also include 4 integrated bipolar leads for EMG, ECG, VEOG and HEOG.

III. METHODS

A. Grouping Phase

Before raw data pre-processing, the 60 EEG channels were grouped into 13 regions (see TABLE I), in order to perform an overall spacial analysis of the different brain regions. This grouping step was done through the average of the signals corresponding to the same region.

TABLE I: EEG channels grouping per regions

Region Considered	**EEG Channels Grouped**
Frontal **(F)**	Fp1, Fpz, Fp2, AF3, AF4
Left Frontal **(LF)**	F7, F5, F3, F1, FT7, FC5, FC3, FC1
Central Frontal **(CF)**	Fz, FCz
Right Frontal **(RF)**	F2, F4, F6, F8, FC2, FC4, FC6, FT8
Left Temporal **(LT)**	T7, TP7
Left Central **(LC)**	C5, C3, C1, CP5, CP3, CP1
Central **(C)**	CPz, Cz
Right Central **(RC)**	C2, C4, C6, CP2, CP4, CP6
Right Temporal **(RT)**	T8, TP8
Left Parietal **(LP)**	P7, P5, P3, P1, PO7, PO5, PO3
Central Parietal **(CP)**	PZ, POz
Right Parietal **(RP)**	P2, P4, P6, P8, PO4, PO6, PO8
Occipital **(O)**	O1, Oz, O2

B. Preprocessing

In the filtering stage of the region signals, a Notch filter was performed for powerline interference removal at 50 Hz. Additionally, it was applied a 4th order Butterworth high pass filter with cut-off frequency at 0.5 Hz for trend removal and a 7th order Butterworth low pass filter with cut-off frequency at 90 Hz, since the frequencies of interest are lower than this limit.

Afterwards, the FastICA algorithm [14] was applied to the filtered signal for Blind Source Separation (BSS) and artefact removal. After identification of possible artefact components, they were removed and a reconstruction of the signals was performed.

C. Feature Extraction

Twelve features were extracted from the EEG signals, using a 5-seconds window with an overlap of 80%. Regarding the time domain, the features extracted were: the signal average, the normalized signal average and the signal variance. On the other hand, in the frequency domain, the features extracted were: the average frequency, the total power and the relative power on five frequency bands: Delta (0-4 Hz), Theta (4-8 Hz), Alpha (8-13 Hz), Beta (13-30 Hz) and Gamma (30-90 Hz). The latter one, Gamma band, since it has a wide range, was divided into 3 subbands: Low Gamma (30-50 Hz), Medium Gamma (50-70 Hz), High Gamma (70-90 Hz).

The time domain features were chosen due to its easy extraction, while the frequency domain features calculated were selected for being the most used ones on EEG signals analysis in emotion recognition areas [15], among others.

D. Baseline Normalization

After the extraction of the above mentioned features for each of the five events and for each trial, a normalization of the features of the Code Comprehension Task was performed, by dividing each feature by the mean value of the correspondent feature computed during the Reading Task and the 2nd Cross. This was performed in order to achieve subject specific baseline invariance.

E. Transformed Features Calculation

In order to capture the state of the subject for each code complexity, but at the same time to maintain sufficient instances for classification, eight parameters were computed from the normalized features for each one quarter of Code Comprehension Task. The parameters calculated were maximum, minimum, mean, standard deviation, variance, median, skewness and kurtosis.

F. Feature Selection and Classification

Through a Leave-One-Subject-Out cross-validation procedure, i.e. for each run the train dataset corresponds to the samples of 29 subjects while the test dataset corresponds to the samples of one subject, filter-based feature selection and dimension reduction, followed by a classifier was implemented.

From the total 1248 features (13 regions $\times$ 12 features $\times$ 8 parameters), was used the robust and noise tolerant filter method ReliefF for feature selection, ranking the features by weights [16]. From this step, was then selected the first two hundred features corresponding to the ones with the higher weights, and eliminated the redundant correlated features through the calculation of Pearson correlation coefficient.

Afterwards, Principal Component Analysis (PCA) was used for dimension reduction of the features previously selected, choosing only the components with a eigenvalue greater than one, according to the Kaiser criterion [17]. We performed an initial feature ranking just to illustrate to the reader which of the features are most discriminant. However, it should be noted that in practice this step is not required as PCA is able to perform a feature space reduction.

After feature selection and dimensionality reduction, for a more general and less computational cost classification, the Linear Discriminant Analysis Classifier was used.

IV. Results and Discussion

Regarding the complexity metrics values, in Fig. 2, it is possible to observe the three types of code complexity evaluation, for each one of the five complexity metrics used. According to the metrics, differences between the three different codes complexity are visible, being the major difference noticed for the McCabe Cyclomatic Complexity metric, one of the most popular and widely used in software engineering [18].

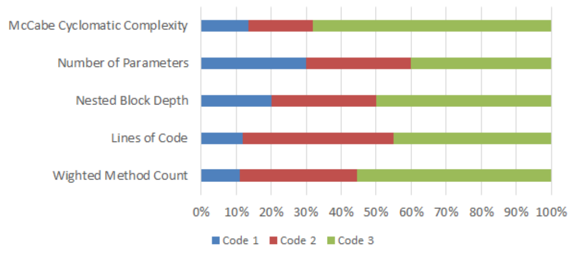


Fig. 2: Complexity level of each Code tasks according to each one of the five software complexity metrics used.

Concerning the results of the NASA TLX Survey, the participants mental effort is evaluated in a scale from 1 to 6, being 1 the minimum effort and 6 the maximum effort. The mean and standard deviation values of the results are presented in the Fig. 3. It is visible that the participants did not feel such great difference between Code 2 and Code 3, while there is a significant difference between the more complex Codes (2 and 3) and Code 1. A first interpretation of this results is that software complexity metrics alone are not an accurate indicator of code complexity as perceived by programmers.

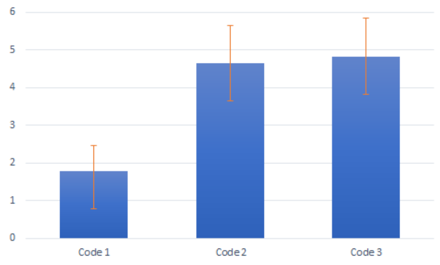


Fig. 3: Results of mental effort felt by the 30 participants according to the answers in NASA TLX survey, for the three types of code comprehension.

For an analysis of the selected two hundred features of each one of the thirty folds, the ones that were common to all folds were chosen. From these features subset, the relative frequency (in %) of each type of the correspondent global feature in the total subset was computed, as represented in the TABLE II. The relative power of Gamma band represents more than 42% of the features in the final subset, being the higher values for the low Gamma (16%) and high Gamma (16%) intervals. In second place, the most frequent features were the signal amplitude variance with 12% and the relative theta power with 8%. These results, mostly regarding gamma features, are in accordance with the literature and some studies on cognitive workload and memory [12], [19], [20], since during code comprehension tasks, it is required high concentration, memory, information processing, among other cognitive functions that are strongly related to higher frequencies, as Gamma frequency [10].

TABLE II: Feature's Relative Frequency

Global Feature Type	Relative Frequency (%)
Relative Power Delta	5.33
Relative Power Theta	8.00
Relative Power Alpha	5.33
Relative Power Beta	5.33
Relative Power Low Gamma	16.00
Relative Power Medium Gamma	10.67
Relative Power High Gamma	16.00
Total Power	6.67
Average Frequency	6.67
Signal Average	2.67
Signal Variance	12.00
Normalized Signal Average	5.33

Another analysis was carried out, with the same approach as above, but regarding the 13 regions considered. The calculation of the relative frequency of regions (in %) that contributed to the set of the common features of the thirty folds was computed. The results can be visualized in the topographic map in Fig. 4. About 64.94 % of features from the final subset of features belong to the parietal (LP, CP and RP regions), occipital (O region) and right temporal lobes (RT region), which is in accordance with the literature, since these are regions related to cognition, visual perception and language comprehension [21]. The right parietal region (RT) was the one with higher relative frequency of features, with a value of 16.88%, which goes in line with one of the regions reported on others studies regarding programming language comprehension and regions activation [12].

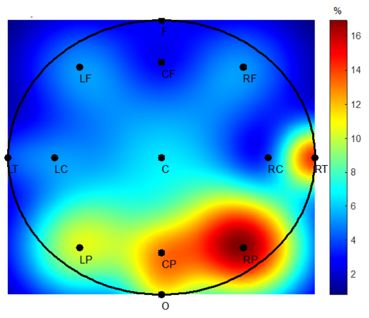


Fig. 4: Topographic map of the frequency (in %) of the most features selected in the 13 regions considered.

Finally, TABLE III presents the classification results using the mentioned Leave-One-Subject-Out procedure. Mean and standard deviation of five considered evaluation parameters,

TABLE III: Evaluation Parameters

Task Classes	Evaluation Parameter (%)				
	Recall	**Specificity**	**Precision**	**F-Measure**	**Accuracy**
Code 1	92.50 ±20.92	97.67 ±7.49	94.22 ±19.40	92.71 ±19.28	70.00 ±15.72
Code 2	59.17 ±38.00	77.04 ±21.95	49.42 ±34.01	52.25 ±33.65	
Code 3	58.33 ±36.75	78.75 ±18.90	52.87 ±32.10	53.13 ±31.32	

for the three classes of code complexity, using the Linear Discriminant Analysis Classifier are presented. From these results, we can observe a good performance in distinguishing the code 1 from the more complex codes 2 and 3. However, it is difficult to differentiate between code 2 and code 3, which goes in line with the subjective mental effort felt by the participants evaluated by NASA-TLX, as shown in Fig.3, but contrary to the expected metric complexity values (see Fig. 2). It should also be mentioned that a high variance can be observed in the classification results. This might be due to the fact that some participants did not made the adequate effort to understand each code. This was particularly observed in the eye tracking data and has to be further investigated.

Globally, these results suggest that EEG based approaches can be considered to introduce biofeedback from programmers in the software development process, as code areas that cause high mental effort will tend to be more error prone. This opens an interesting research path, using software programming as a paradigm for intellectual tasks that could benefit with the idea of human biofeedback.

V. CONCLUSIONS

In this paper we assessed cognitive stress during code comprehension tasks using EEG and compare multiple standard features to standard tools used in software engineering, i.e., complexity metrics, as well as the NASA-TLX tool which is commonly applied in mission critical environments. It can be concluded that is possible to distinct with high confidence simple code (code 1) from more complex code (code 2 and 3) using EEG. However code 2 (middle complexity level code) and code 3 (highest complexity level code) were not well distinguished, what suggests that the features only discriminate highly different tasks complexity, revealing an evidence of saturation with the complexity level, being coherent with the answers of the NASA-TLX survey. Nevertheless, the results did not match the evaluations of the codes using the complexity metrics. This suggests that software complexity metrics do not capture cognitive load and therefore do not translate a key element in bug production - the human element. Therefore, additional research is required to complement current software engineering strategies in order to integrate the programmer in the loop.

ACKNOWLEDGMENT

The authors acknowledge the financial support of the national project BASE (POCI - 01-0145 - FEDER- 031581).

REFERENCES

[1] J. Castelhano, I. C. Duarte, C. Ferreira, *et al.*, "The role of the insula in intuitive expert bug detection in computer code: an fmri study," *Brain imaging and behavior*, pp. 1–15, 2018.

[2] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer: an investigation of how developers spend their time," in *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*. IEEE Press, 2015, pp. 25–35.

[3] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[4] M. H. Halstead *et al.*, *Elements of software science*. Elsevier New York, 1977, vol. 7.

[5] V. Gruhn and R. Laue, "Complexity metrics for business process models," in *9th international conference on business information systems (BIS 2006)*, vol. 85. Citeseer, 2006, pp. 1–12.

[6] R. Shriram, M. Sundhararajan, and N. Daimiwal, "Eeg based cognitive workload assessment for maximum efficiency," *Int. Organ. Sci. Res. IOSR*, vol. 7, pp. 34–38, 2013.

[7] C. P. Niemic and K. Warren, "Studies of emotion," *A Theoretical and Empirical Review of Psychophysiological Studies of Emotion.(Department of Clinical and Social Psychology). JUR Rochester*, vol. 1, no. 1, pp. 15–19, 2002.

[8] J. Siegmund, C. Kästner, S. Apel, *et al.*, "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] N. Peitek, J. Siegmund, S. Apel, *et al.*, "A look into programmers' heads," *IEEE Transactions on Software Engineering*, 2018.

[10] M. V. Kosti, K. Georgiadis, D. A. Adamos, *et al.*, "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.

[11] I. Crk, T. Kluthe, and A. Stefik, "Understanding programming expertise: an empirical study of phasic brain wave changes," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 23, no. 1, p. 2, 2016.

[12] S. Lee, A. Matteson, D. Hooshyar, *et al.*, "Comparing programming language comprehension between novice and expert programmers using eeg analysis," in *2016 IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE)*. IEEE, 2016, pp. 350–355.

[13] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[14] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[15] S. M. Alarcao and M. J. Fonseca, "Emotions recognition using eeg signals: A survey," *IEEE Transactions on Affective Computing*, 2017.

[16] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[17] H. F. Kaiser, "The application of electronic computers to factor analysis," *Educational and psychological measurement*, vol. 20, no. 1, pp. 141–151, 1960.

[18] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software engineering*, vol. 26, no. 8, pp. 797–814, 2000.

[19] U. Friese, M. Köster, U. Hassler, *et al.*, "Successful memory encoding is associated with increased cross-frequency coupling between frontal theta and posterior gamma oscillations in human scalp-recorded eeg," *Neuroimage*, vol. 66, pp. 642–647, 2013.

[20] T. F. Quatieri, J. R. Williamson, C. J. Smalt, *et al.*, "Using eeg to discriminate cognitive workload and performance based on neural activation and connectivity," MIT Lincoln Laboratory Lexington United States, Tech. Rep., 2016.

[21] S. Fitzgibbon, K. Pope, L. Mackenzie, *et al.*, "Cognitive tasks augment gamma eeg power," *Clinical Neurophysiology*, vol. 115, no. 8, pp. 1802–1809, 2004.