Cognitive Flexibility and Programming Performance

Marianne Leinikka

Finnish Institute of Occupational Health
Helsinki, Finland

marianne.leinikka@ttl.fi

Arto Vihavainen

Department of Computer Science
University of Helsinki, Finland

arto.vihavainen@helsinki.fi

Jani Lukander

Finnish Institute of Occupational Health
Helsinki, Finland

jani.lukander@ttl.fi

Satu Pakarinen

Finnish Institute of Occupational Health
Helsinki, Finland

Faculty of Behavioral Science
University of Helsinki, Finland

satu.pakarinen@ttl.fi

Keywords: POP-V.A. Cognitive Theories, POP-IV.F. Exploratory

Abstract

Cognitive flexibility is an integral part in adaptive behavior such as learning, and can be seen as one explaining factor in performance in various tasks. In this study, we explore the possibilities of a dynamic psychological test that can be administered online to assess cognitive flexibility and apply the test across a student population that is learning to program. While our results imply that cognitive flexibility has little correlation with the score of a traditional pen-and-paper programming exam as well as the students' average grade from their first semester of studies, cognitive flexibility does play a role in the efficiency with which students solve programming errors. Moreover, our results imply that while cognitive flexibility correlates with the students' efficiency in solving programming errors, the correlation is more evident with novice programmers than with programmers with existing experience.

1. Introduction

Cognitive flexibility, the ability to switch between modes of thought and to simultaneously think about multiple concepts, is an essential part of adaptive behavior, such as flexible everyday functioning and learning (Boger-Mehall 1996). While the underlying processes such as attention and short-term memory evolve over age from infants to old age (Anguera et al. 2013, Crone et al. 2006) and vary considerably between individuals of same age and culture (Daneman and Carpenter 1980), the cognitive performance also varies as a consequence of stress (Olver et al. 2014, Aggarwal et al. 2014), fatigue and sleep deprivation (Fogt et al. 2010, Sallinen et al. 2013), as well as environmental disturbances such as interruptions and background noise (Banbury and Berry 1998, Reynolds et al. 2013).

As learning has become a lifelong task, measuring cognition has become increasingly popular for promoting the productivity and well-being of learning, working, and aging citizens. Cognitive flexibility has been traditionally measured with neuropsychological tests such as the Trail Making test (Reitan 1955, Reitan 1958) and The Stroop Test (Stroop 1935), which are, however, targeted for detecting rather gross developmental or acquired impairments in clinical conditions, and require extensive expertise and resources that are limited in the healthcare and educational systems. On the other hand, more sensitive and more specific cognitive tests for evaluating mild cognitive

impairments or cognitive processes in normal population exist, but those are developed for research purposes and require the acquisition of expensive exclusive software platforms.

Another way to measure cognitive flexibility is by means of task-switching paradigms. Several variations of the task-switching paradigms exist (Dibbets and Jolles 2006, Gupta et al. 2009, Kopp and Lange 2013), but in general, participants are put to work in a controlled environment, where frequent but random shifts between two tasks are required. The paradigm is based on two main principles: first, people are faster and more accurate when performing a task repeatedly; this phenomenon is called *repetition benefit*, and second, when changing the task or switching between tasks one's performance becomes slower and less accurate/more erroneous; described with the *switch cost* and *mixing cost* (for a review on task-switching, see Monsell 2003).

When considering computer programming, a considerable amount of effort has been invested in identifying factors and creating tests that can be used to seek out students who e.g. struggle with learning to program. While some discussions on the cognitive aspects exist (Parnin 2010), typical approaches include programming aptitude tests (Evans and Simkin 1989, Tukiainen 2002), tests that measure skills such as mathematical and spatial reasoning (White and Sivitanides 2003, Fincher et al. 2006), and the use of different self-perception metrics to detect students likely to succeed or fail (Bergin and Reilly 2006, Ramalingam et al. 2004). With the improved computing facilities, more opportunities have arisen due to the ability to gather data from students' programming process, making it possible to capture and analyze programming sessions (Jadud 2006, Rodrigo et al. 2009, Watson et al. 2013).

In this work, we contribute to the understanding of the link between programming performance and cognitive flexibility, as well as provide researchers with a tool for measuring cognitive flexibility. Our study implies that while results from a cognitive flexibility test do not correlate with traditional course outcome metrics such as the result from a pen-and-paper exam, cognitive flexibility plays a role in the efficiency with which students solve programming errors. Moreover, cognitive flexibility correlates more strongly with the performance of novice programmers than with the performance of more experienced programmers, suggesting that programming experience influences the results.

The remainder of the paper is structured as follows. Next section discusses the research methodology, target groups, and the system used in more detail, after which we discuss the test results and provide a detailed analysis. In the fourth section, we discuss the limitations of our study, and finally, in the fifth section, we conclude with an overview of the work and outline future work.

2. Research Method and Materials

Our study consists of two main questions, where we first identify whether the task-switching test that is administered online performs comparably with existing task-switching studies. Once the first question has been investigated, we seek for connections between task-switching performance and programming performance.

RQ 1: Are the results from the task-switching game comparable with existing task-switching studies?

RQ 2: What correlation is there between task-switching results and programming performance? More explicitly, how do the task-switching results correlate with...

RQ 2.1: … error-solving velocity?

RQ 2.2: … programming course exam scores?

RQ 2.3: … overall performance during the first semester of studies?

The study is based on two datasets from courses offered by the University of Helsinki. The first dataset is from a programming course targeted for University students that was held during fall 2013, while the second dataset is from an open programming course from spring 2014 that was offered for free for everyone. While the University students who participated in the study completed the game in a class with a teacher and received a small chocolate bar as a token of appreciation for participation,

the participants from the open programming course have had no additional motivating factors for completing the game outside the course material asking them to participate in ongoing research.

For the RQ 2.1., we use snapshot data gathered from both programming courses, while for RQ 2.2 and RQ 2.3, snapshot data from only the course offered for the University students is used as there is no formal exam procedure in the open programming course. Linear regression analysis is performed on the results from the task-switching study and the snapshot data used in each research question.

2.1 Test Procedure

The task that is used for this study is a modified three-phased version of a Number-Letter task (Vandierendonck et al. 2010, Pashler 2000). The test utilizes a defined subset of Arabic numbers (2-9) and Latin letters (a, e, g, i, k, m, r, u) that are commonly used in the Anglo-American, German and Fenno-Ugric language groups. Letter-number pairs (e.g. a7) are presented to the participant either above or below a stationary horizontal line, and to avoid the possibility of the participants fixating on a certain location a horizontal jitter is used. The responses are given from the keyboard by pressing either the X or M key. The test has three tasks, and at the end of each task, the percentage as well as the mean value of reaction time for correct answers is shown to the participant.

In the first task, the detection task, a baseline *reaction time* is measured by requesting the subject to press key X as soon as they see a stimulus on screen. The detection task consists of 10 trials. The letter-number pairs are visible until the response, but for a maximum of 1950 ms. Button presses 40-1950 ms post-stimulus onset are accepted as responses. The responses up to 39 ms post-stimulus onset are discarded from the data to discount possible mis-hits.

The second task measures *categorization*. First, letter-number pairs are presented above the horizontal line and the task is to categorize the number of the pair as either odd or even by pressing X or M, respectively. Thereafter, stimulus pairs are presented below the horizontal line and the task is to categorize the letter of the pair as either consonant or vowel by pressing X or M, respectively. The letter-number pairs are quasi-randomized so that an equal amount of each letter and number will appear but in randomized pairs of all possible combinations. The categorization task consists of 80 trials, first 40 above and then 40 below the line.

Finally, in the third task, the *task-switching* task, the participant performs both the categorization tasks alternatingly. The letter-number pairs are presented alternatingly above or below the horizontal line. The task changes according to the location cue: when the letter-number pair is presented above the horizontal line, the task is to classify shown numbers as odd or even, and when the pair is presented below the horizontal line, the task is to classify the letter of the pair as consonant or vowel, by pressing either X or M, respectively.

The task-switching task consists of 168 stimulus pairs with 63 task switches. 1 to 6 stimulus pairs are presented consecutively below or above the line before a switch. There are 23 trials consisting of a single pair before switch, 10 trials constituting 2, 3, and 4 repetitions before switch, each, and 5 trials of 5, and 6 repetitions, each. The number of repetitions before switch was pseudorandomized.

In the Categorization and the Task-switching tasks, the letter-number pairs are presented until the participant's response, but for a maximum of 2500 ms. The rate of stimulus presentation is tied to the subjects responses in the following way: The successive stimulus is presented 150 ms after a correct response, and 1500 ms after an incorrect or missed response. Key presses occurring between 40 ms and 2500 ms from the stimulus onset are accepted as responses.

The participants are instructed that both speed and accuracy in the task are equally important. The participants are obliged to briefly practice each of the tasks (excluding the Detection task) at least once but a maximum of three times before the actual task. Each practice round contains six letter-number pairs. As our task-switching task has been accommodated for classroom use, it is notably shorter when compared to the ones used in previous studies, and lasts 7-10 minutes on average.

2.2 System Description

For the participants, the test is provided as an online test that can be linked e.g. from course material or via an email. Source code, hosting instructions, and an online version are available at http://github.com/measureself/cognitive-flexibility

When considering the system architecture, the front-end of the system has been developed as a single-page web application that can be used both in mobile phones and desktop computers (the mobile version requires an additional application in the mobile phone that acts as a wrapper in order to accommodate touch-screen user interfaces). The application has been built using modern web technologies (HTML5, CSS3 and JavaScript), and it uses jQuery to avoid cross-platform issues. The system has been designed to be easily extendable, and provides easy access points for changing the language, alphabet and input configuration used, as well as for changing other configuration parameters such as the response windows.

The backend implementation of the system is abstracted behind a REST-API, which means that the storage implementation can be changed if deemed necessary without modifications to the front-end. All data is transferred using JSON, and the existing backend has been implemented using Java EE and Spring Framework. In addition to Spring Framework, EclipseLink is used to abstract the data layer, making it easier to use different databases. Currently, by default, the system uses a MySQL-server as the database.

When asking users to take the test, one can either provide the users with an URL that contains the desired username, which automatically will then be used for identification, or, users can be asked to sign into the system via a front page. The authentication mechanism provided by the backend is modular, and additional mechanisms such as support for a third party authentication service can be added in a straightforward fashion. In the current version, the system only stores the desired username and related results, and does not ask for any additional information. Hence, mapping demographic information should be done separately, or added to the system if needed.

In order to measure reaction times accurately, the system utilizes high resolution time support whenever available in the browser. The *Performance.now()*-interface provided by the high resolution time support gives the browser the access to the current time in sub-millisecond precision, which has previously been impossible. Naturally, we acknowledge that the system that the participant has, and the load on participants' systems also affects the timing. This is partially solved by gathering information on the user's system (user agent, window size), but, in the current version, no feasible means for measuring the load on the user's machine exist.

3. Evaluation and Discussion

For evaluation, a total of four studies are conducted. In the first study, we verify that the results obtained from the system are compatible with existing task-switching studies, and in the three subsequent studies we investigate the correlations between the task-switching performance and (1) error solving velocity, (2) programming course exam scores, and (3) credits and average grade from the first semester.

3.1 Are the results from our system comparable with existing task-switching studies?

To answer RQ 1, a total of 298 participants completed the entire task-switching game. Of those, 15 participants were excluded from the data due to performing at or below the level of 75% (Monsell 2003) and five participants due to missing demographic information. Of the remaining 278 participants (mean 25.7 years, $\sigma = 8.6$) 39 were female (mean 25.1 years, $\sigma = 5.4$). Tables 1 and 2 present the mean values and the standard deviations for the Detection task and the Categorization task and the corresponding values for the Task-Switching task. The hit rates, false responses, and reaction times in the task-switching task as a function of stimulus pair repetitions after switch are shown in Figure 1.

	Detection task	Categorization task
Hits (%)	96.3, σ = 17.4	94.7, σ = 5.2
False responses (%)	-	5.1, σ = 5.1
Misses (%)	3.7, σ = 17.4	1.1, σ = 0.5
RT for correct answers (ms)	434.4, σ = 141.4	640.2, σ = 74.4
RT for incorrect answers (ms)	-	628.6, σ = 137.0

Table 1 – The mean values and the standard deviations of key parameters for the Detection and the Categorization task.

	Task-switching task Stimulus pair position after switch					
	1	2	3	4	5	6
Hits (%)	87.3, σ = 9.4	92.1, σ = 9.1	94.6, σ = 6.9	95.6, σ = 8.4	93.1, σ = 11.1	95.1, σ = 17.1
False responses (%)	9.4, σ = 8.1	5.5, σ = 7.5	4.1, σ = 5.8	3.4, σ = 7.5	6.3, σ = 10.6	4.1, σ = 15.1
Misses (%)	3.2, σ = 4.5	2.5, σ = 5.3	1.3, σ = 3.2	1.0, σ = 3.6	0.6, σ = 3.3	0.7, σ = 6.0
RT for correct answers (ms)	1193.3, σ = 179.7	863.3, σ = 158.4	776.4, σ = 127.9	778.2, σ = 156.6	793.1, σ = 168.4	796.4, σ = 228.4
RT for incorrect answers (ms)	1160.7, σ = 396.3	1111.9, σ = 458.9	964.6, σ = 399.6	890.0, σ = 376.1	862.3, σ = 347.0	731.5, σ = 325.6

Table 2 – The mean values and the standard deviations of key parameters for the Task-switching task as a function of stimulus pair position after switch.

Within the task-switching task, the reaction time for correct responses immediately after the switch (position 1) differed from the mean RT for repetitions (hits for the stimulus pairs in positions 3-6): $t277 = 50.2$, $P < 0.001$. Similarly, the reaction time for correct responses at position 2 after the switch differed from the mean RT for repetitions (positions 3-6): $t277 = 13.6$, $P < 0.001$. This switch cost was on average 424 ms (σ = 141 ms), i.e., 56.4% for position 1 and 94 ms (σ = 115 ms), i.e., 12.6% for position 2 after the switch.

When comparing the Task-switching task and the Categorization task, the mean reaction time for hits after the switch (mean of positions 3-6) differed from the reaction time for hits in the Categorization task: $t277 = 21.4$, $P < 0.001$. This mixing cost was on average 129 ms (σ = 100 ms), i.e., 20.4% for the mean of positions 3-6 after the switch.

As expected, switching between tasks results in poorer accuracy (decreased hit rate) and slower response (increased reaction time; Figure 1). This effect is strongest immediately after switch and decreases with task repetition: the switch cost for the stimulus position 1 after the switch was 56.4% and for position 2, 12.6%. The decrease in reaction time already between the stimulus pair immediately after switch (position 1) and the pair following it (position 2) suggests that the studied participants are able to recover fast from the switch and inhibit the irrelevant task rule.

Even though repetition of the same task results in participants being faster and more accurate the more the task is repeated in succession the performance in the Task-switching task never reaches the level of speed and accuracy in the Categorization task. This mix cost for the mean positions 3-6 was 20.4%, indicating that despite repetition the categorization was more difficult within the Task-switching task

as compared to the simple Categorization task. The benefit of repetition is also seen in the proportion of false responses, i.e., the error rate. The error rate was higher immediately after the switch compared to the stimulus pair in positions 3-6 (9.4%, σ = 8.1 and 4.3%, σ = 5.0, respectively; Figure 1). The results are in accordance with the previous studies (Monsell 2003).

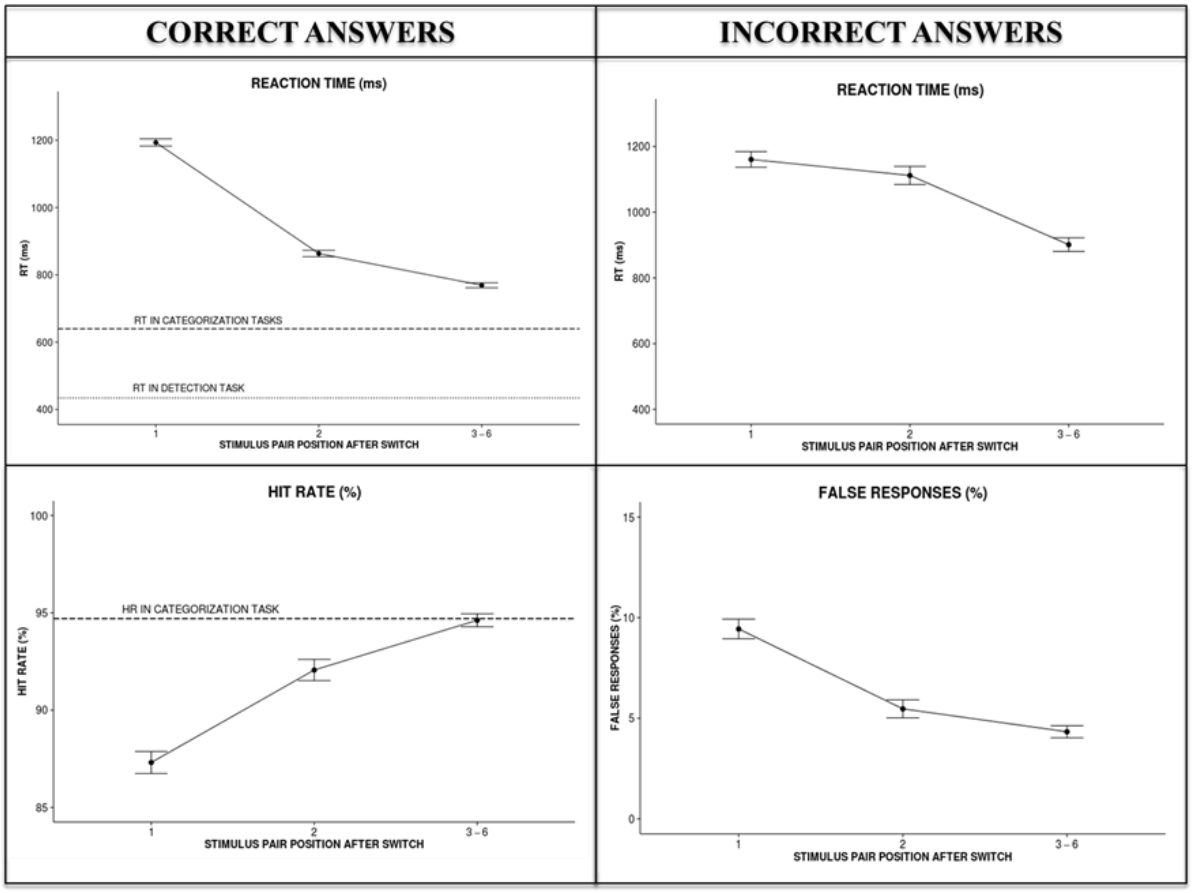

Figure 1 – Reaction times, hits and false responses of 278 participants as a function of stimulus pair repetitions after switch in the Task-switching task. The upper panels in Figure 1 represent the mean reaction times for correct (left panel) and incorrect (right panel) responses as a function of stimulus pair position (1-6) after switch. The lower panels represent the proportion of correct (left panel) and incorrect (right panel) answers as a function of stimulus pair position (misses not shown). Error bars denote the standard error of mean (S.E.M.). For comparison, the mean reaction time (RT) in the Detection task (dotted line) and in the Categorization task (dashed line) are presented in the upper left panel. Also the proportion of correct answers in the Categorization task (dotted line) is shown in the lower left panel.

3.2. Are the task-switching results correlated with programming performance?

To answer the RQ 2.1; "Are the task-switching results correlated with performance in solving programming errors?" we investigate key-level snapshot data that has been extracted as students program in the programming courses. The snapshots are from participants that have worked on a problem where the goal is to create a data repository for bird observations. The problem is open ended, which means that the structure of the student's solution is not helped or enforced in any way, and the criterion for success is that the student's application fulfills the functionality that is provided as the following textual input-output description (inputs from the user are denoted with italics) with additional notes on the different commands needed.

```
? Add
Name: Raven
Latin Name: Corvus Corvus
? Add
Name: Seagull
Latin Name: Dorkus Dorkus
? Observation
What was observed:? Seagull
? Observation
What was observed:? Turing
Is not a bird!
? Observation
What was observed:? Seagull
? Statistics
Seagull (Dorkus Dorkus): 2 observations
Raven (Corvus Corvus): 0 observations
? Show
What? Seagull
Seagull (Dorkus Dorkus): 2 observations
? Quit
```

We consider compilation errors in the snapshots (e.g. incorrect type, missing class, missing semicolon) as programming errors, and utilize the median time that it takes for the participant to solve compilation errors as the measure of performance. Participants have been asked about their programming background, and participants have been split into two categories based on their answers.

In the dataset with data from students that have worked on the bird repository problem as well as answered the task-switching questionnaire, there are 45 participants with no programming experience, while 65 participants have previous programming experience.

	r (Pearson) no programming experience, N=45	*r (Pearson)* existing programming experience, =65
RT in Detection task	.18	.18
RT in Categorization task	.39**	.25*
Switch cost for position 1	-.32*	-.12
Switch cost for position 2	-.21	-.09
Total RT for Task-Switching task	.54**	.30*
RT for correct answers in Task-Switching task	.57**	.32**
RT for repetition in Task-Switching task	.57**	.36**
Mixing cost	.47**	.16
RT-Task-Switching-minus-RT-Detection	.35*	.12

(* $p < 0.05$, ** $p < 0.01$)

Table 3 –Pearson product moment correlation coefficients for various aspects of task-switching and time taken by participants to solve programming errors, classified by programming experience.

In Table 3, we can observe that the correlations between the median problem solving times are stronger in the population that have had no previous programming experience before the start of the course; the task switching correctness shows moderate correlation with the median speed for solving a programming error, while other indicators show low to no correlation. For the population that had previous programming experience, the correlations, when existing, are low, as their results are more heavily influenced by their programming experience. As the reaction time in the detection task does not correlate with the participants' problem solving speed, we can assume that the correlations are not explained by reaction time.

Next, we will investigate RQ 2.2 “Are the task-switching results correlated with performance in programming course exams?” and RQ 2.3 “Are the task-switching results correlated with overall performance in the first semester?”. For the course exam performance, we observe the points that students have received in a written paper exam (0-30), while for the overall performance in the first semester, we look at the number of study credits and average grade that the students have received during their first semester.

In the dataset used for RQ 2.2 and RQ 2.3, we have received the study records from 45 students who both participated in the task-switching study during their first semester, and agreed to their information being used for the study.

	Performance in programming course exam	Number of study credits received during the first semester	Average grade during the first semester
	r (Pearson)	*r (Pearson)*	*r (Pearson)*
RT in Detection task	-.11	.15	-.15
RT in Categorization task	-.15	.05	-.13
Switch cost for position 1	.08	-.08	.12
Switch cost for position 2	.06	-.05	.01
Total RT for Task-Switching task	-.12	-.09	-.32*
RT for correct answers in Task-Switching task	-.14	-.05	-.34*
RT for repetition in Task-Switching task	-.19	-.02	-.38*
Mixing cost	-.13	-.08	-.34*
RT-Task-switching-minus-RT-Detection	-.01	-.16	-.15

(* $p < 0.05$)

Table 4 – Pearson product moment correlation coefficients for various aspects of task-switching and performance in a written programming course exam, number of study credits received during the first semester, and the average grade during the first semester.

In Table 4 we can observe that the task-switching performance has no correlation with the performance in the course exam and the amount of credits that the students receive during the first semester. However, low negative correlation can be observed between the average grade from the first semester and the third task-switching task. One plausible explanation for the negative correlation, while hypothetical at this point, is that students with higher cognitive flexibility have learned in their previous studies (e.g. in high-school) that they do not have to work too hard to succeed at some level. If they have entered their university studies with the same expectation, it is possible that the lack of effort can be visible in the first semester grades.

4. Limitations of Study

Our assumption for the study was that the majority of study participants come from a homogeneous population (i.e. the participants are university students and people interested in learning programming). When considering the previous programming experience, variance in existing computing skills was not taken into account outside the reported programming experience. This reduces the internal validity as it is possible that other attributes such as the tendency to play computer games may explain some of the results. In our current study, we assume that both populations have people who also play computer games. As it is possible that this is not the case, we wish to stress that

one should seek for additional background variables; in our case the obvious "hindsight" was that we should have asked the participants to elaborate on their programming experience and whether they spend time playing e.g. computer games. Similarly, although cognitive flexibility varies over age, we did not consider the participants' ages in the study, as we did not focus on students with learning impairments.

Another limitation is that our data comes from two teaching contexts (local, online) that both follow the same pedagogy and material (for details on the pedagogy, see Vihavainen et al. 2011). While every teacher has their own approach to teaching programming, an objects-early -approach is used at the University of Helsinki. In addition, the students are guided in local labs, and students tend to form study groups before the exams to study "exam-related" -topics. That is, the correlations may differ when the study is replicated in another teaching context. However, correlations from all tests were reported, which makes replication easier.

We assume that some variance will be visible depending on the teaching approach, language used, tools used, and naturally the student population. However, we sought to remedy parts of this by using linear regression for fitting the models. While this may have led to losing some insight that could have been gained with e.g. polynomial regression, this deliberately avoids overfitting and thus our results can be used as a baseline for future investigation.

5. Conclusions and Future Work

We have described initial steps in measuring cognitive flexibility using an online task-switching test, and sought out correlations between different aspects in learning to program. Cognitive flexibility was chosen over other measures such as IQ as it provides a lens to abilities which may be related to computer programming; the ability of switching between modes of thought and the ability of thinking about multiple concepts simultaneously. We first verified that the results from the task-switching game are comparable with existing studies, after which we sought correlations between task-switching performance and programming performance. While the correlation between the performance in solving programming errors and the task-switching performance was significant, the outcome of a paper exam and first-semester studies had practically no correlation with the task-switching performance.

We acknowledge that the conditions and teaching context, such as the programming language, pedagogy, and different support mechanisms used differ considerably across different teaching contexts. While we have taken initial steps to create understanding on how cognitive flexibility affects programming performance, further work is needed to determine the applicability of the task-switching test across different teaching contexts. Similarly, comparing its efficiency with other tests is needed.

As the system created for this study is published as an open-source project that is free-of-charge and also relatively device independent, gathering larger datasets from students in different contexts is straightforward. The system improves the existing opportunities on evaluating the effects of different demographic variables such as sex, age, cultural background and education not just on programming performance, but on performance in other learning contexts as well. On the basis of the test one can also draw a learning curve on successful repetitions of the test. Effects of fatigue, stress, or the time of day can be assessed by comparing different sessions from the same individuals or if the data set is sufficiently large, even as a cross-sectional study of different individuals, provided that information on their stress or time-of-day when taking the test is available for instance via a web-based survey.

Our work leaves several questions open. Possible directions that require effort from multiple parties include investigating whether the observations hold in (1) different teaching contexts (e.g. online, lab-centric environment, traditional classroom environment), (2) different populations, (3) how the existing predictors such as programming aptitude tests and IQ tests relate to task-switching performance, and (4) how successive repeated test sessions are visible in both the test results and possibly other factors.

6. Acknowledgements

This research has been funded by the Tekes project #337910202.

7. References

Aggarwal, N. T., Wilson, R. S., Beck, T. L., Rajan, K. B., de Leon, C. F. M., Evans, D. A., & Everson-Rose, S. A. (2014). Perceived Stress and Change in Cognitive Function Among Adults 65 Years and Older. Psychosomatic medicine, 76(1), 80-85.

Anguera, J. A., Boccanfuso, J., Rintoul, J. L., Al-Hashimi, O., Faraji, F., Janowich, J., ... & Gazzaley, A. (2013). Video game training enhances cognitive control in older adults. Nature, 501(7465), 97-101.

Banbury, S., & Berry, D. C. (1998). Disruption of office-related tasks by speech and office noise. British Journal of Psychology, 89(3), 499-517.

Bergin, S., & Reilly, R. (2005, June). The influence of motivation and comfort-level on learning to program. In Proceedings of the PPIG (Vol. 17, pp. 293-304).

Boger-Mehall, S. R. (1996). Cognitive flexibility theory: Implications for teaching and teacher education. In Society for Information Technology & Teacher Education International Conference (Vol. 1996, No. 1, pp. 991-993).

Crone, E. A., Wendelken, C., Donohue, S., van Leijenhorst, L., & Bunge, S. A. (2006). Neurocognitive development of the ability to manipulate information in working memory. Proceedings of the National Academy of Sciences, 103(24), 9315-9320.

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. Journal of verbal learning and verbal behavior, 19(4), 450-466.

Dibbets, P., & Jolles, J. (2006). The switch task for children: Measuring mental flexibility in young children. Cognitive Development, 21(1), 60-71.

Fogt, D. L., Kalns, J. E., & Michael, D. J. (2010). A comparison of cognitive performance decreases during acute, progressive fatigue arising from different concurrent stressors. Military medicine, 175(12), 939-944.

Gupta, R., Kar, B. R., & Srinivasan, N. (2009). Development of task switching and post-error-slowing in children. Behavioral and Brain Functions, 5(38), 1-13.

Jadud, M. C. (2006, September). Methods and tools for exploring novice compilation behaviour. In Proceedings of the second international workshop on Computing education research (pp. 73-84). ACM.

Kopp, B., & Lange, F. (2013). Electrophysiological indicators of surprise and entropy in dynamic task-switching environments. Frontiers in human neuroscience, 7.

Monsell, S. (2003). Task switching. Trends in cognitive sciences, 7(3), 134-140.

Olver, J. S., Pinney, M., Maruff, P., & Norman, T. R. (2014). Impairments of Spatial Working Memory and Attention Following Acute Psychosocial Stress.Stress and Health.

Parnin, C. (2010). A cognitive neuroscience perspective on memory for programming tasks. In the Proceedings of the 22nd Annual Meeting of the Psychology of Programming Interest Group (PPIG).

Pashler, H. (2000). 12 Task Switching and Multitask Performance. Control of cognitive processes, 277.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In ACM SIGCSE Bulletin (Vol. 36, No. 3, pp. 171-175). ACM.

Reitan, R. M. (1955). The Relation of the Trail Making Test to Organic Brain Damage1. Group, 3(1), 1.

Reitan, R. M. (1958). Validity of the Trail Making Test as an indicator of organic brain damage. Perceptual and motor skills, 8(3), 271-276.

Reynolds, J., McClelland, A., & Furnham, A. (2013). An investigation of cognitive test performance across conditions of silence, background noise and music as a function of neuroticism. Anxiety, Stress & Coping, (ahead-of-print), 1-12.

Rodrigo, M. M. T., Baker, R. S., Jadud, M. C., Amarra, A. C. M., Dy, T., Espejo-Lahoz, M. B. V., ... & Tabanao, E. S. (2009). Affective and behavioral predictors of novice programmer achievement. ACM SIGCSE Bulletin, 41(3), 156-160.

Sallinen, M., Onninen, J., Tirkkonen, K., Haavisto, M. L., Härmä, M., Kubo, T., ... & Porkka-Heiskanen, T. (2013). Effects of cumulative sleep restriction on self-perceptions while multitasking. Journal of sleep research, 22(3), 273-281.

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. Journal of experimental psychology, 18(6), 643.

Tukiainen, M., & Mönkkönen, E. (2002). Programming aptitude testing as a prediction of learning to program. In Proc. 14th Workshop of the Psychology of Programming Interest Group (pp. 45-57).

Vandierendonck, A., Liefooghe, B., & Verbruggen, F. (2010). Task switching: interplay of reconfiguration and interference control. Psychological bulletin,136(4), 601.

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011). Extreme apprenticeship method in teaching programming for beginners. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 93-98). ACM.

Watson, C., Li, F. W., & Godwin, J. L. (2013). Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. In Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on (pp. 319-323). IEEE.

White, G., & Sivitanides, M. (2003). An empirical investigation of the relationship between success in mathematics and visual programming courses. Journal of Information Systems Education, 14(4), 409-416.