Assessing software design skills and their relation with reasoning skills

Dave R. Stikkolorum[1], Claire E. Stevenson[2], and Michel R.V. Chaudron[3]

[1]Leiden Institute of Advanced Computer Science, [2]Department of Psychology
Leiden University, The Netherlands
[3]Joint Department of Computer Science and Engineering, Chalmers University of Technology and Gothenborg University, Sweden
[1]drstikko@liacs.nl, [2]cstevenson@fsw.leidenuniv.nl, [3]chaudron@chalmers.se

Abstract. Lecturers see students struggle learning software design. In order to create educational interventions it is needed to know which (abstract) reasoning skills are related to students' software design performance. We introduce an online software design skills test for measuring students' software design skills. Two student groups of two different European universities participated in an experiment in where we were able to relate students' visual and verbal reasoning skills to students' software design skills and measured learning improvement. In the future proper interventions can be chosen while using the test as a diagnostic tool.

Keywords: reasoning, software design, assessing, education, UML

1 Introduction

Lecturers from all over the world see students struggle with the subject of software design. Not only syntactic errors are made when using modeling languages like UML, but also semantic or organization (design) errors. Kramer argues that the key lies in students' abstract reasoning[7]. The objective of our research is to discover which reasoning skills are related to the design skills of software engineering students. We focus on two types of abstract reasoning: visual and verbal reasoning. In our study the main question is: 'Which type of knowledge and/or reasoning skills are related to students' software design skills?' This leads to the following underlying questions: RQ_1 - Can verbal or visual reasoning ability predict ones software design skills? RQ_2 - Do language skills influence software design skills? RQ_3 - Does prior domain knowledge (UML) influence software design skills and learning? Answering these questions can help lecturers to create educational interventions. In order to measure students' software design skills we developed a test. As far as we know there is no standard measurement instrument of software design skills. In this paper we analyze two groups of students at two different universities. They participated in a series of tests addressing software design, modeling, reasoning and language skills. In section 2 we describe related work. In section 3 we describe our method. The results are presented in section 4 and discussed in section 5. We conclude and propose future work in section 6.

2 Related work

Several researchers have discussed the importance of subjects that should be included in the curricula of university software engineering programs [5] [6]. Especially inclusion of mathematics is subject of discussion. Lethbridge found that software professionals remembered little mathematics from their study programs[8]. Some use this research to state that curricula emphasise mathematics too much while others, like Henderson use this as an argument to claim not to trust professionals' opinions[4], because there is too little research on the effect of mathematics on software engineering skills. In our study we aim to identify what general reasoning skills (not only mathematical) are related to performance on software design. Bennedsen and Caspersen studied abstraction as indicator for students' learning performance on software engineering [1]. They were not able to find evidence for this relationship. Roberts [12] found positive correlation between abstraction ability and course grades, but observed a small number of students (N=15). We targeted a larger group of students, included language knowledge and used our test as main indicator of students' design ability.

3 Method

In this section we explain the research method employed to develop our instrument for measuring software design skills. We wanted the measure to show an increased score after students had followed a course on software design. Therefore, we asked students to perform the test at the start (pretest) of a course and at the end (posttest) of a course. We found subjects for our test through two different courses on software design taught at two different universities in Northern Europe. We presented our design skills test as additional learning material.

In this section we describe our hypotheses. We address the participants and discuss the different types of test instruments that we used.

3.1 Hypotheses

In all hypotheses we focus on the effect of the independent variables on the level of design skills (dependent variable), shown in table 1. The level of design skills is measured at two points in time: with a pretest and with a posttest. The hypotheses we want to examine are: H_1 - UML domain knowledge will not influence students' design skills. H_2 - Visual reasoning is related to design skills test performance. H_3 - Verbal reasoning is related to design skills test performance. H_4 - Knowledge of the English Language (language of our design skills test) is related to test performance.

3.2 Participants and Data Collection

The students that participated in the test were 2nd year BSc. students from two universities in Europe. A group from Chalmers University in Gothenburg -

Hypothesis	Construct	Description	Type of variable
1	UML Knowledge	UML syntax knowledge	Independent
2	Visual Reasoning	Raven figure series	Independent
3	Verbal Reasoning	Verbal analogies	Independent
4	Knowledge of English	C-Test for languages	Independent
all	Design Skills Pretest	Software Design Skills	Dependent
all	Design Skills Posttest	Software Design Skills	Dependent

Table 1. Measured Constructs

Sweden and a group from Utrecht University in Utrecht - The Netherlands. Both groups had no or very little experience with software design. The initial number of students(N) was 243, however not all students participated on all tests during their course. For some parts of the analysis we had to use a smaller number of students.

All data was collected with on-line multiple choice tests[1]. This was convenient for assessing a larger group of participants. We used an open-source questionnaire tool called LimeSurvey[2].

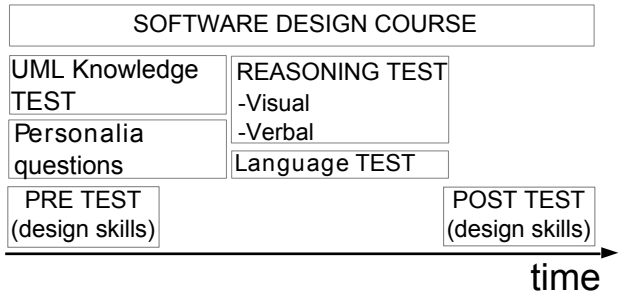


Fig. 1. Test construction in time dimension

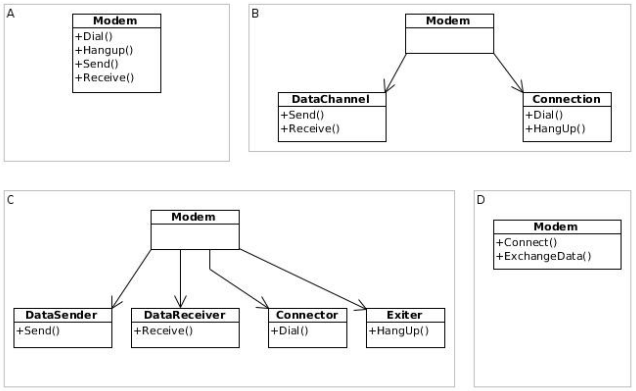
3.3 Designed Procedure

In figure 1 the organization of the test is shown in the dimension of time. The whole experiment consists of 6 test parts: design skills pre- and posttest, UML Knowledge, Reasoning, Language and one part that is about personal information. The experimental procedure was as follows: 1) In the first week students were administered the software design pretest, the UML prior knowledge test and answered general questions about age, background and experience. 2) In the next weeks they followed the software design course at their university and

[1] A demo will be made available at: http://umltest.liacs.nl
[2] http://www.limesurvey.org

were asked to complete the verbal and visual reasoning tests. Also their level of English was tested in these weeks. 3) At the end of the course the students made a software design skills posttest.

Pre and Post software design skills tests The pre- and posttest both consisted of 20 similar multiple choice items targeting software design principles such as mentioned in [11] and [9] with a time limit of 40 minutes. In some questions the student is asked to compare different designs for the same system. An example question is shown in figure 2. In other questions only one design was



Which one is a better design, considering assignment of responsibility?

Please choose only one of the following:

- ○ Design A, because the system is too small to split up in different classes with different responsibilities.
- ○ Design B, because operations that are part of the same task are combined to a responsibility.
- ○ Design C, because every operation is a responsibility.
- ○ Design D, because it is necessary to reduce the amount of operations in a class, not the responsibility.

Fig. 2. example question design skills test

presented and students had to answer questions about this design. The designs were presented to the students in the Unified Modeling Language (UML[3]). The UML is the most popular modeling lanuage at the moment of writing. We choose a very small subset of the UML for the reason that we only see the UML as a vehicle for designing software systems. Lecturers and Phd students discussed about the possible answers. Only the questions where they agreed on the answers were elected. The cognitive difficulty levels we used are up to level two of Bloom's taxonomy of educational objectives [14].

UML prior Knowledge A set of 22 items about UML syntax knowledge was administered after the pretest to be able to study the relationship between prior UML knowledge and design skills afterwards. There was a 20 minutes time limit.

[3] http://www.uml.org

Language and Reasoning tests We identified three possible types of knowledge and/or skills that could be related to software design skills: Language knowledge, Verbal Reasoning and Visual Reasoning. In order to study the relationship between the performance on the design skills test we asked the subjects to make a test that measures these skills. For the Language knowledge we used the automated C-test for languages of Leuven University[4]. For Verbal Reasoning we used a verbal analogies test[5], for visual reasoning we used a test based on Raven's progressive matrices [10]. The time limit was 60 minutes.

Personalia A couple of questions were asked after the first test about prior design experience, education and other pre-knowledge.

4 Results

In this section we describe the results of the individual test instruments. The analysis of this data will be discussed in section 5. We show psychometric properties, descriptive statistics, investigate correlations and compare the universities performance. The student groups from the universities are anonymized and shown as 'A' and 'B' or we consider the groups as a total.

4.1 Psychometric Properties

We used classical test theory to determine reliability of our instruments. Cronbach's α coefficient of internal consistency was .44 for the pretest, .58 for both the posttest and UML knowledge test. The α is somewhat low because of measuring different knowledge constructs. The item difficulty (i.e., proportion correct) was lower for the pretest (M=.59, SD=.17, range=.21-.82) than the posttest (M=.68, SD=.17, range=.25-.89). For the UML knowledge test the students solved on average 41% of the items correctly (M=.41, SD=.25, range=.09-.90).

4.2 Descriptive Statistics

Table 2 shows the number (N) of students that participated per test, Minimum (Min) and Maximum (Max), Mean (M), standard deviation (SD), the Skewness (Skew) and Kurtosis (Kurt). We excluded students' responses if they responded to only less than 50 percent of the questions on a test.

4.3 Correlations between instruments and linear regression

Figure 3 shows the Pearson correlations that were found between the individual tests. A correlation coefficient of .10 is seen as a weak relationship, .30 as moderate, and 0.5 as a strong relationship [2]. Figure 3 show a significant ($p < .01$)

[4] http://www.arts.kuleuven.be/ctest/english
[5] http://www.fibonicci.com/verbal-reasoning/analogies-test

Construct	N	Min	Max	M	SD	Skew	Kurt
Design Skills Pre	243	3	19	11.73	2.75	-.31	-.03
UML Knowledge	217	2	19	9.11	3.12	-.09	-.21
Visual Reasoning	177	0	18	13.27	2.80	-1.41	4.24
Verbal Reasoning	173	0	15	9.05	3.06	-.55	-.12
English language	155	0	38	25.31	8.08	-1.31	1.86
Design Skills Post	171	5	19	13.41	3.00	-.44	-.15

Table 2. Descriptive Statistics Test Instruments

moderate relationship ($r = .377$) between visual reasoning and the design skills posttest. This also counts for verbal reasoning and the posttest ($r = .380$, $p < .01$). The visual and verbal reasoning tests do not have this relationship with the design skills pretest. The English language test does not seem to correlate with other tests. There is a moderate to strong relationship between the verbal and visual reasoning tests. Also the design skills pre- and posttest have a moderate strong ($r = .434$, $p < .01$) correlation. We found a moderate correlation between posttest and the exam of university A ($r = .317$) and a strong correlation between posttest and the exam of university B ($r = .536$) both at significant level of .01.

Correlations between test instruments

		UML Knowledge	Visual Reasoning	Verbal Reasoning	English Language	Design Skills Post	Exam A	Exam B
Design Skills Pre	Pearson Correlation	,276**	,230**	,180*	,11	,434**	,14	,327**
	Sig. (2-tailed)	,00	,00	,02	,21	,00	,12	,00
	N	217	162	158	141	159	133	80
UML Knowlegde	Pearson Correlation		,09	,01	,02	,09	,03	,373**
	Sig. (2-tailed)		,29	,89	,80	,31	,75	,00
	N		145	142	130	140	122	68
Visual Reasoning	Pearson Correlation			,490**	,12	,377**	,12	,311*
	Sig. (2-tailed)			,00	,13	,00	,26	,01
	N			173	155	134	87	61
Verbal Reasoning	Pearson Correlation				,303**	,380**	,18	,337**
	Sig. (2-tailed)				,00	,00	,10	,01
	N				155	131	86	60
English language	Pearson Correlation					,186*	,03	,06
	Sig. (2-tailed)					,05	,82	,67
	N					116	80	50
Design Skills Post	Pearson Correlation						,317**	,536**
	Sig. (2-tailed)						,01	,00
	N						74	70

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Fig. 3. Correlations between the individual test instruments

A series of linear regression models were used to investigate which factors (pretest, verbal reasoning, visual reasoning, UML knowledge or English language proficiency) best predicted the student's posttest performance. The best fitting parsimonious model explained 34% of variance ($F(3, 121)=122.36$, $p<.001$) and is represented by posttest = β_{pre} • pretest + β_{vis} • visual reasoning + β_{verb}

• verbal reasoning. With β_{pre}=.40, $t_{pre} = 5.27$, $p_{pre} < .001$; β_{vis}=.14, $t_{vis} = 1.63$, $p_{vis} = .11$ and β_{verb}=.25, $t_{verb} = 2.99$, $p_{verb} < .01$

4.4 Comparison between universities

We compared the performance of all instruments between university A and B. We found significant differences between universities the UML Knowledge test and the C test. University A performed better on the C test ($M_A = 27.06$, $SD_A = 8.2$, $M_B = 24.11$, $SD_B = 7.9$, $p = .03$). University B performs better on the UML knowledge test ($M_A = 8.3$, $SD_A = 3.03$, $M_B = 9.8$, $SD_B = 3.04$, $p = 0.00$).

5 Discussion

The correlation coefficients show that both verbal and visual reasoning explain almost 40 percent of the performance on student's design skills posttest. This is in contrast with the correlation of these skills with design skills pretest. This indicates abstract reasoning contributes to improvement of software design skills($H_{2,3}$). We did not use a control group. One could argue improvement of skills is due retesting and not due learning. The correlation between the posttest and the exam scores provides evidence the we measure learning improvement. We measured the reasoning abilities of students with tests that measure general abilities and considered not trainable. With this insight we maybe have to focus on the matter how problems are presented or lectured for those that do not have this natural 'talent' for abstract reasoning. The fact that both the UML knowledge and language test had no correlation with the design skills pretest and posttest($H_{1,4}$) indicates that we indeed succeeded in questioning design concepts and not about UML problems. Also the fact that university B performed better on the UML knowledge test while both universities not performed significantly different on the design skills pretest provides further support. The students achieved higher scores on the design skills posttest than on the design skills pretest. This indicates that they learned during the course.

6 Conclusions and Future Work

In this paper we presented our findings of an on-line test for measuring software design skills and abstract reasoning skills of students. We showed the relationship between abstract reasoning and the ability of solving software design problems. Although abstract reasoning on its own can not be trained, educational intervention in the future can be explored with the focus on how we teach and not per se what we teach. We believe approaches where experienced based or game based learning is used can help. We already gained positive feedback on a pilot of our motivational game 'The Art of Software Design'[6][3][13]. We plan to extend the game with the findings of this experiment. In the future, indicated by

[6] http://aosd.host22.com

our regression model, lecturers can use our test to diagnose students and choose appropriate interventions when educating software design students.

Acknowledgments

We would like to thank the students and lecturers from Gothenburg University and Utrecht University for their participation in this study.

References

1. Jens Bennedssen and Michael E. Caspersen. Abstraction ability as an indicator of success for learning computing science? In *Proceedings of the Fourth international Workshop on Computing Education Research*, ICER '08, pages 15–26, New York, NY, USA, 2008. ACM.
2. J. Cohen. *Statistical power analysis for the behavioral sciences*. Erlbaum, 1988.
3. Oswald de Bruin. The art of software design, creating an educational game teaching software design, 2012.
4. Peter B. Henderson. Mathematical reasoning in software engineering education. *Commun. ACM*, 46(9):45–50, September 2003.
5. Peter B. Henderson. Math counts: Mathematical reasoning in computing education. *ACM Inroads*, 1(3):22–23, September 2011.
6. Peter B. Henderson. Mathematical reasoning in computing education ii. *ACM Inroads*, 2(1):23–24, February 2011.
7. Jeff Kramer. Is abstraction the key to computing? *Communications of the ACM*, 50(4):36–42, April 2007.
8. T.C. Lethbridge. What knowledge is important to a software professional? *Computer*, 33(5):44–50, 2000.
9. RC Martin. Design principles and design patterns. *Object Mentor*, (c):1–34, 2000.
10. John Raven. The raven's progressive matrices: change and stability over culture and time. *Cognitive psychology*, 41(1):1–48, 2000.
11. Arthur J. Riel. *Object-Oriented Design Heuristics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1996.
12. Patricia Roberts. Abstract thinking: a predictor of modelling ability? 2009.
13. Dave R. Stikkolorum, Michel R.V. Chaudron, and Oswald de Bruin. The art of software design, a video game for learning software design principles. In *Gamification Contest MODELS Innsbruck*, 2012.
14. Lorin W Anderson, David R Krathwohl, Peter W Airasian, Kathleen A Cruikshank, Richard E Mayer, Paul R Pintrich, James Raths, and Merlin C Wittrock. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Abridged Edition*. Allyn & Bacon, 2000.