Studies into interactive didactic approaches for learning software design using UML

Stikkolorum, D.R.

Part II

Software Design Comprehension

Chapter 2

An Instrument for Measuring Design Skills

Description of contributions of the authors
The research in this chapter was done in collaboration with Claire Stevenson. At the time of writing, Claire was Ph.D. student in Psychology at the faculty of social science at Leiden University, where she specialised in the topic of statistics for testing. At the same time she studied for the bachelors program in computer science at Leiden University. As part of this bachelor program, she did a project-study on this topic in collaboration with Dave Stikkolorum. Claire introduced us to the tests that measure abstract reasoning and helped us to perform statistical analyses and interpret the results. The writing of the publication on which this chapter was based was a joint effort.

The contributions of Dave Stikkolorum were: creating the design skill test, running the experiment, and collaborating on the analyses and interpretation.

Michel Chaudron acted as supervisor to both Dave and Claire on this project.

In this chapter we present an instrument for assessing software design skills. In order to create educational interventions for teaching software design we need to know which reasoning skills are related to students' software design performance.

This chapter is based on the following publication: Dave R. Stikkolorum, Claire Stevenson, and Michel R.V. Chaudron. Assessing software design skills and their relation with reasoning skills. In *EduSymp 2013. CEUR, vol. 1134, paper 5*, 2013

We introduce an online test for measuring students' software design skills and relate those to abstract reasoning. Two student groups of two different European universities participated in an experiment in which 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.

2.1 Introduction

Lecturers from all over the world see students struggle with the subject of software design. Not only syntactic errors are made when using modelling languages like UML, but also semantic or organisation (design) errors. Kramer argues that the key lies in students' abstract reasoning skills [69]. One 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 this study the main question is:

RQ$_{main}$ 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 a student's 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 measurement instrument of software design skills. In this chapter we analyse two groups of students at two different universities. They participated in a series of tests addressing software design, modelling, reasoning and language skills. The remainder of this chapter is organised as follows: in Section 2.2 we describe related work. In Section 2.3 we describe our method. The results are presented in Section 2.4 and discussed in Section 2.5. We conclude and propose future work in Section 2.6.

2.2 Related Work

Several researchers have discussed the importance of subjects that should be included in the curricula of university software engineering programs [52] [53]. Especially inclusion of mathematics is a subject of discussion. Lethbridge found that software professionals remembered little mathematics from their study programs[77]. 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[51], 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 ability as indicator for students' learning performance on software engineering [14]. They were not able to find evidence for this relationship. Roberts [113] found positive correlation between abstraction ability and course grades, but observed a small number of students (N=15). We target a larger group of students (N=243), included language knowledge and used our test as main indicator of students' design ability.

2.3 Method

In this section we explain the research method employed to develop our instrument for measuring software design skills. We want the measure to show an increased knowledge of software design after following 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.

2.3.1 Hypotheses

In all hypotheses we focus on the effect of the independent variables on the level of design skills (dependent variable), shown in Table 2.1. The level of design skills is

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 2.1: *Measured Constructs*

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 design skills test performance.

2.3.2 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 - Sweden and a group from Utrecht University in Utrecht - The Netherlands. Both groups had no or very little experience with software design at the start of the course. The initial number of students(N) was 243, however not all students participated in all tests during their course. For some parts of the analysis we had to use a smaller number of students (N=155 was the minimum). This was due students that left the course prematurely or due technical issues. Section 2.4.2 shows Table 2.2 with the descriptive statistics of the test instruments.

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].

[1] A demo is available at: https://designskillstest.drstikko.nl
[2] https://www.limesurvey.org

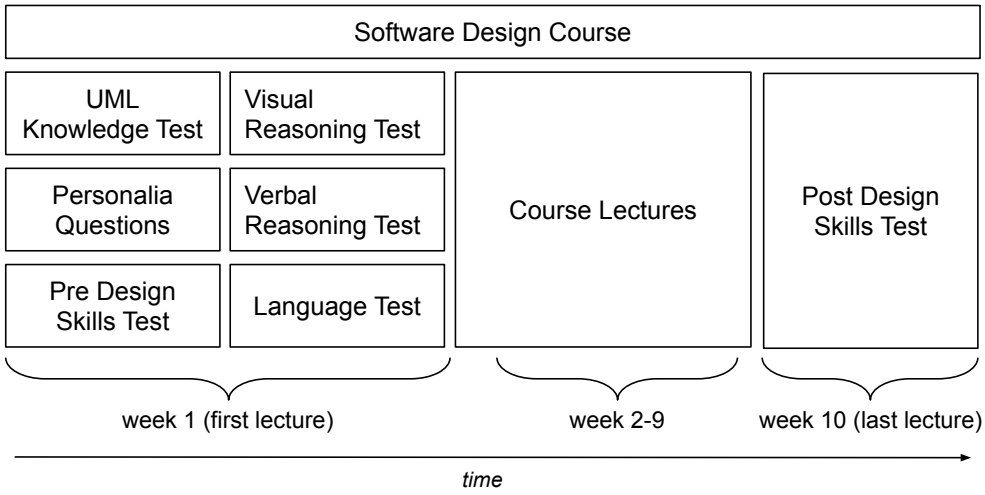

Figure 2.1: *Test construction in time dimension*

2.3.3 Experiment Design

In Figure 2.1 the organisation 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, Abstract Reasoning, Language Skills 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 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 the software design skills posttest.

Pre and Post software design skills tests The pre- and posttest both consisted of 20 similar multiple choice questions about software design principles such as mentioned in [112] and [88] with a total time limit of 40 minutes. In some questions the student is asked to compare different designs for the same system. An example of such a question is shown in figure 2.2. In other questions only one design was presented and students had to answer questions about this design. The designs were presented to the students in the Unified Modelling Language (UML[3]). UML is the most popular software modelling language at the moment of writing. We choose a very small subset of UML for the reason that we only see UML as a vehicle for designing software

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

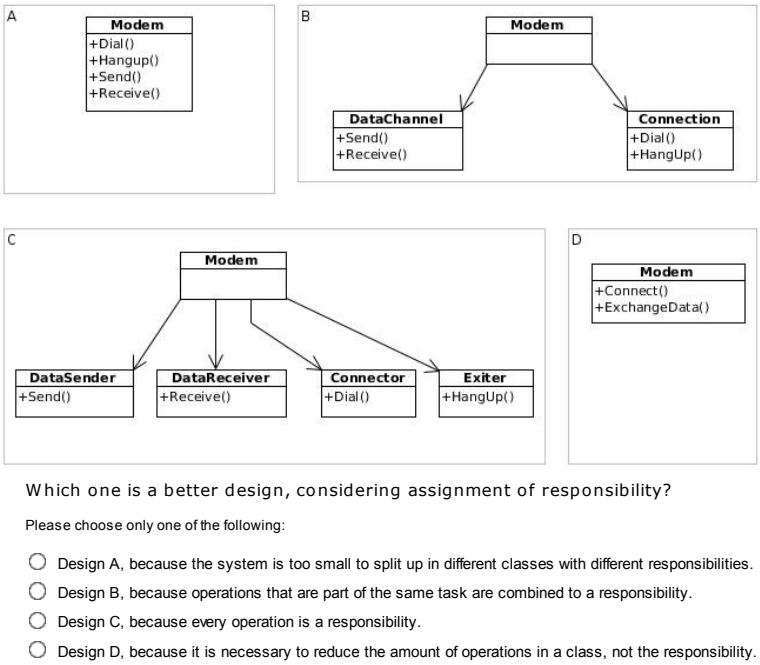

Figure 2.2: *Example question from the design skills test*

systems. Lecturers and PhD students discussed about the possible answers. Only those questions were elected, where they agreed on the answer. The cognitive difficulty levels we used are up to level two of Bloom's taxonomy of educational objectives [159]: 'understanding'.

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 for this UML test.

Language and Reasoning tests We focused on three possible types of knowledge and/or skills that could be related to software design skills: English 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 from Leuven University[4][12]. For verbal reasoning we used a verbal analo-

[4]http://www.arts.kuleuven.be/ctest/english - not available anymore

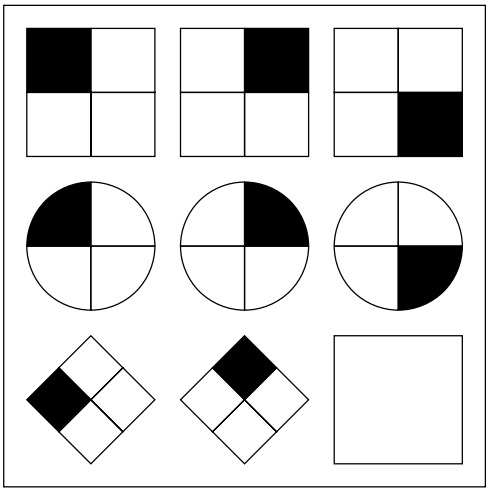

Figure 2.3: *Raven test example question*

.... stands to book as painter stands to ....

1. Chapter	2. Writer	3. Reading	4. Word	5. Literature
A. Painting	B. Picasso	C. Brush	D. Workshop	E. Paint

Figure 2.4: *Verbal analogies example question*

gies test[5], for visual reasoning we used a test based on Raven's progressive matrices [110]. Example questions can be seen in Figures 2.3 and 2.4. The time limit was 60 minutes.

Personalia After the first test students were asked five questions about their personal situation. Topics were: gender, age, prior software design experience, education and work experience.

2.4 Results

In this section we describe the results of the individual test instruments. The analysis[142] of this data will be discussed in section 2.5. We show psychometric properties, descrip-

[5] https://www.fibonicci.com/verbal-reasoning/analogies-test

tive statistics, investigate correlations and compare the universities' performances. The student groups from the universities are anonymized and shown as 'A' and 'B' or we consider the groups as a total.

2.4.1 Psychometric Properties

We used classical test theory to determine reliability of our instruments. Cronbach's α coefficient of internal consistency was 0.44 for the pretest, 0.58 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=0.59, SD=0.17, range=0.21-0.82) than the posttest (M=0.68, SD=0.17, range=0.25-0.89). For the UML knowledge test the students solved on average 41% of the items correctly (M=0.41, SD=0.25, range=0.09-0.90).

2.4.2 Descriptive Statistics

Table 2.2 shows the number (N) of students that participated per test, Minimum (Min) and Maximum (Max) score, Mean (M) score, standard deviation (SD), the Skewness (Skew) and Kurtosis (Kurt). We excluded students' responses if they responded to less than 50 percent of the questions on a 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 2.2: *Descriptive Statistics Test Instruments*

2.4.3 Correlations Between Instruments and Linear Regression

Figure 2.5 shows the Pearson correlations that were found between the individual tests. A correlation coefficient of 0.10 is considered as a weak relationship, 0.30 as moderate, and 0.5 as a strong relationship [28]. Figure 2.5 show a significant ($p < 0.01$

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).

Figure 2.5: *Correlations between the individual test instruments*

) moderate relationship ($r = 0.377$) between visual reasoning and the design skills posttest. This also counts for verbal reasoning and the posttest ($r = 0.380$, $p < 0.01$). 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 ($r = 0.490$, $p < 0.01$). Also the design skills pre- and posttest have a moderate to strong ($r = 0.434$, $p < 0.01$) correlation. We found a moderate correlation between posttest and the exam of university A ($r = 0.317$) and a strong correlation between posttest and the exam of university B ($r = 0.536$) both at significant level of 0.01. 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 = $\beta_{pre} \bullet$ pretest + $\beta_{vis} \bullet$ visual reasoning + $\beta_{verb} \bullet$ verbal reasoning. With $\beta_{pre}=.40$, $t_{pre} = 5.27$, $p_{pre} < .001$; $\beta_{vis}=.14$, $t_{vis} = 1.63$, $p_{vis} = .11$ and $\beta_{verb}=.25$, $t_{verb} = 2.99$, $p_{verb} < .01$

2.4.4 Comparison Between Universities

We compared the performance of all instruments between the universities. We found significant differences between the scores on the UML Knowledge test and the C-test. University A performed better on the C-test ($M_A=27.06$, $SD_A=8.2$, $M_B=24.11$, $SD_B=7.9$, $t(153)=2.27$, $p=.03$). University B performed better on the UML test ($M_A=8.3$, $SD_A=3.03$,

M_B=9.8, SD_B=3.04, t(215)=3.57, p=0.00).

2.5 Discussion

The correlation coefficients show that both verbal and visual reasoning explain almost 40 percent of the performance on the students' design skills posttests. This is in contrast with the correlation of these skills with the 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 to retesting and not due to learning. The correlation between the posttest and the exam scores provides evidence the we measure learning improvement. We used tests that are considered not trainable. They measure students' abstract intelligence. This means we have to investigate the specific sub tasks related to abstract intelligence or how problems are presented during lectures 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 UML notation problems. Also the fact that university B performed better on the UML knowledge test while both universities did not perform significantly different on the design skills pretest provides further support for trusting our test is mostly independent from knowledge of UML notation. The students achieved higher scores on the design skills posttest than on the design skills pretest. This indicates that they have learned design skills during the course.

2.6 Conclusions and Future Work

In this chapter 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 intelligence cannot be trained, we see challenges and opportunities in exploring educational interventions for specific reasoning tasks and/or alternative teaching methods. We believe game based learning could be used. This is motivated in our other research in game based learning (Chapter 3). We already gained positive feedback on a pilot of our motivational game 'The Art of Software Design'[6][31][133]. We plan to extend the game with the findings of this experiment.

In the future, indicated by our regression model, lecturers can use our test to diag-

[6] https://gamejolt.com/games/the-art-of-software-design/21373

nose students and choose appropriate interventions when educating software design students.