Johanyák, Zs. Cs., Pap-Szigeti, R. and Alvarez Gil, R. P.: Analyzing students' programming failure, A GAMF Közleményei, Kecskemét, XXII. (2008), ISSN 0230-6182, pp. 115-120.
http://johanyak.hu

Analyzing students' programming failures

Dr. Johanyák, Zsolt Csaba[1] – Pap-Szigeti, Róbert[2] – dr. Alvarez Gil, Rafael Pedro[3]

This paper reports the results of a research done at Kecskemét College to examine information engineering students' typical programming failures. A questionnaire-based measurement was organized for this purpose. The expectations regarding the positive influence of the visual methods were not fulfilled. The utilization of the student feedback in order to help the students in better understanding of the thought material as well as in the development of their self-concept is considered extremely important. The results confirm that the level of practicing and understanding is very important in successful teaching.

1. Introduction

Software engineering is a special discipline that is similar to other engineering disciplines in many aspects, but basically defines different tasks [2]. Although in the last decades several software development methodologies and approaches have been published one can often read or hear from software project failures (schedule slips, buggy releases, missing features, system crashes, etc.).

Programming is a complex and difficult task. The customer usually does not know exactly at the beginning what she/he wants and later she/he continuously changes her/his requirement. Thus software engineers should possess solid methodological grounds as well as strong knowledge of programming languages and development tools. Teaching software engineering has a vital role in the acquirement of these skills.

We teach software engineering for information engineering students at Kecskemét College. Although programming is not the primary topic in their curriculum many of them will be hired as a software developer. Thus the effectiveness of the software development teaching is especially important. As

[1] PhD, associate professor, Kalmár Sándor Institute of Information Technology, Kecskemét College
[2] Senior lecturer, Kalmár Sándor Institute of Information Technology, Kecskemét College
[3] PhD, associate professor, Kalmár Sándor Institute of Information Technology, Kecskemét College

part of our continuous curriculum development and improvement process we examine periodically [1] the typical programming failures of our students.

This paper reports the results of a research targeting three main topics. The first one covered the occurrence frequency of some typical programming failures by students who have not enrolled yet on the visual programming course. The second topic we was interested in was whether the skills and knowledge acquired in course of the visual programming and RAD (Rapid Application Development) training had a positive influence regarding the failures. Finally we examined the typicality of the students' failures experienced by us during the laboratories.

The rest of this paper is organized as follows. Section 2 presents briefly our software engineering education. Section 3 demonstrates a questionnaire-based experiment about students' programming failures.

2. Software engineering teaching at Kecskemét College

The IT students of Kecskemét College attain the basic programming skills in three semesters. In course of the *Problem Classes, Algorithms* course unit placed in the first semester, they go deeply into the concepts of algorithms and they get acquainted with the related description methods. The basics of C programming language are taught during the second semester. This unit is called *Programming I.* In the next semester comes the *Programming II.* course unit that continues the training of the C language extended with non object-oriented elements of C++. Relevant topics are data structures, file input-output operations, functions, dynamic memory management, etc. The students have to take an exam at the end of the course. The *Programming Paradigms and Techniques* unit comes in the next semester. Its syllabus comprehends the basics of Object-Oriented Programming (OOP) using the C++ programming language. It should be mentioned that the students are trained in database management parallel with the programming courses in the second and third semesters.

The *Programming Paradigms and Techniques* unit is followed by the compulsory *Visual Programming* class. The students get acquainted in its frames with the visual application development through the use of a high level development tool (Visual Studio 2008 Professional), which supports RAD techniques. They also learn a new object-oriented language called C#. In parallel with the *Visual Programming* appears the course *Software Engineering*. This is the first time when students familiarize themselves with the different models and methodologies, CASE tools, as well as concepts and construction practice of UML diagrams.

3. Empirical results

3.1. Sample and measuring devices

We organized a questionnaire-based measurement for analyzing our students' programming failures. The sample (target respondents) of our examination consisted of 126 students, who were chosen from students enrolled in the course units *Visual Programming* and *Software Engineering*. 49 of them have begun to learn C# language at the beginning of the semester. The measurement was done at the end of the semester. Approximately one third of the sample contained students who failed their *Programming II.* exams at their first course enrollment.

In order to examine the characteristic programming failures we prepared a questionnaire applying scaled questions (Likert [7]) with ten degrees. The first 18 sentences of the questionnaire asked about the frequency of failures that can be bound equally to the C/C++ and C# languages. Another 14 questions concerned the failures related to C/C++ language only. The last third of the failures dealing with failures typical of the C# language contained 8 items. The reliability of the first two parts (filled in by all students) was acceptably high, $\alpha = 0.89$. The indicator of the reliability was also calculated for the whole questionnaire ($\alpha = 0.89$), although it was filled in by only 49 students.

The rest of the questionnaire studied the background data and some components of programming self-concept. We evaluated the self-concept based on a scale with seven Likert-styled questions. Some sentences with negative meaning were also included in the questions. In those cases the scale of the answers was reversed. Despite the low number of items the scale proved to be reliable ($\alpha = 0.85$); the variables settled into a single factor (KMO = 0.85; Bartlett-test: $\chi^2 = 354.95$; $p < 0.001$). The created factor with % point scale converged to the normal distribution well.

3.2. Failures typical of both C/C++ and C# languages

The mean and the standard deviation was assessed on the scale of the answer (value 1 meant that the student never makes the failure; the answer was value 10 when the student makes the mistake always). The lowest means (1.78) appeared in the case of basic syntax errors (for example "I write mark '<>' in place of mark '!='."). Failures related to complex programs (for example "I do not know what a DLL is and how I can generate and use it.") have the highest mean values (6.00).

We factorized the variables of the first two sub-questionnaires (part I.: KMO = 0.732; Bartlett-test: $\chi^2 = 614.03$; $p < 0.001$; part II.: KMO = 0.759; $\chi^2 = 512.08$; $p < 0.001$). The resulting factors are presented in Table 1.

Table 1. Failure factors for both languages

ID of question	Name of the factor	Mean
1., 2., 13., 14.	Basic syntax errors	2.3
3., 4.	Bracket failures	2.0
5., 6., 7., 11., 12., 16., 17., 18.	Problems of complex types and programs	4.3
22., 23., 24., 25., 27.	Character and string problems	3.2
19., 20., 21., 26., 28.	Structure and pointer failures	4.2
30., 31., 32.	Header file problems	4.5

None of these factors showed significant difference between the two student groups (C#-learners and others). However, one can find lower values by those students, who took the *Programming II.* by the first attempt. The difference is significant in the case of complex variable types ($x_1 = 3.97$; $x_2 = 4.85$; $F = 0.51$; $p = 0.48$; $t = 2.33$; $p = 0.02$) and in the case of programs splitted into more files ($x_1 = 4.08$; $x_2 = 4.79$; $F = 0.84$; $p = 0.36$; $t = 2.19$; $p = 0.03$). It indicates presumably that many students should have much more practice at experimental level to reach higher abstraction levels [3] [4].

3.3. Problems typical of the C# language

As we mentioned before this part of the questionnaire was filled in by only 49 students. The answers were not arranged into well-separable factors (KMO = 0.54). It is worthy to compare the contexts of truth with the factors presented early for the problems appearing in a contact with this language. We calculated the regressions using C#-concerned problems as dependent variables and factors presented early as independent variables. These show that the factors can explain from one seventh to a quarter of the variance of C#-concerned failures (between 12.85% and 29.5%). Therefore it was worthy the independent examination of these failures.

3.4. Motives of our students

The academic achievement can be significantly different among students having similar family backgrounds, intelligence and pre-knowledge. Both the teachers' experiences and the pedagogic research confirm that the motives play a considerable role in the learning performance beside the cognitive components of the psyche [5] [6].

The attitudes for programming were measured in a Likert-styled sub-questionnaire with five degrees. We found a significant difference between the two student groups regarding this component. Surprisingly, the attitude of C#-

learners is significantly lower than other students' attitude ($x_1 = 2.45$; $x_2 = 3.03$; $F = 0.11$; $p = 0.74$; $t = 2.72$; $p = 0.01$). We cannot explain this difference by other variables, because they do not separate by parents' qualifications ($F = 0.12$; $p = 0.73$; $t = 1.65$; $p = 0.10$) and by number of unsuccessful exams in *Programming II.* ($\chi^2 = 1.25$; $p = 0.26$). It is easy to understand that the attitude of those who passed their *Programming II.* exam at the first attempt is significantly higher than others' ($x_1 = 3.07$; $x_2 = 2.21$; $F = 4.57$; $p = 0.03$; $d = 4.47$; $p < 0.001$).

The mean of programming self-concept is 47.2 %p in the whole sample, which shows that our students' self-concept do not reach the medium level. The C#-learners' self-concept is significantly lower than the other students' ($x_1 = 41.8$; $x_2 = 50.7$; $F = 6.70$; $p = 0.01$; $d = 2.34$; $p = 0.02$).

4. Conclusions

As a result of the analysis we got a clearer view of the occurrence frequency of some programming failures considered as typical, which helps us emphasizing certain parts of the thought material.

Surprisingly our expectations regarding the positive influence of the visual methods were not fulfilled. The poor results are understandable at the beginning of studying a new programming language and technique. However, in the long run the lower self-concept can have a negative reaction to the academic achievement. Therefore we consider extremely important to improve the utilization of the student feedback in order to help the students in better understanding the thought material as well as in the development of their self-concept.

Our results confirm that the level of practicing and understanding is very important in successful teaching. Progressing without them can cause loss of motivation and decrease of interest in programming, which could turn into the obstacle of the further successful learning.

REFERENCES

[1] Johanyák, Zs. Cs., Tóth, Gy. F.: Vizuális módszerek oktatásának hatása a hallgatók programozási hibáira, Matematika-, fizika- és számítástechnika oktatók XXXI. konferenciája, Dunaújváros, 2007. augusztus 23-25., pp. 126-131.

[2] Pollice, G.: Teaching software development vs. software engineering, http://www.ibm.com/developerworks/rational/library/dec05/pollice/index.html, 2005.

[3] Piaget, J.: Az értelem pszichológiája. Gondolat Kiadó, Budapest, 1993.

[4] Nagy, J.: XXI. század és nevelés. Osiris Kiadó, Budapest, 2000.

[5] Józsa, K.: Az elsajátítási motiváció és a kognitív kompetencia fejlesztése. In: Csapó Benő és Vidákovich Tibor (ed.): Neveléstudomány az ezredfordulón. Nemzeti Tankönyvkiadó, Budapest, 2001. pp. 162-174.

[6] Csapó, B.: A képességek fejlődése és iskolai fejlesztése. Akadémiai Kiadó, Budapest, 2003.

[7] Likert, R.: A Technique for the Measurement of Attitudes, NY: Archives of Psychology, 140, 1932, pp. 44-53.

A hallgatók programozási hibáinak vizsgálata

Dr. Johanyák Zsolt Csaba – Pap-Szigeti, Róbert, – dr. Alvarez Gil, Rafael Pedro

Összefoglaló

Dolgozatunkban a Kecskeméti Főiskola mérnök informatikus hallgatóinak jellegzetes programozási hibáit vizsgáló kutatásunk eredményeiről számolunk be. A felmérést hallgatók által kitöltött űrlapok segítségével végeztük. Az eredmények igazolják, hogy a begyakorlottság és a megértés szintje fontos szerepet játszik a sikeres oktatásban.

Analyse der Programmierungsfehlers der Studenten

Dr. Johanyák, Zsolt Csaba – Pap-Szigeti, Róbert, – dr. Alvarez Gil, Rafael Pedro

Zusammenfassung

Dieser Beitrag meldet die Resultate einer Forschung, die an der Hochschule Kecskemét erfolgt wurde, um typischen Programmierungsfehlers der Informatikstudenten zu untersuchen. Eine Fragebogen-basierte Datenerhebung wurde zu diesem Zweck organisiert. Die Erwartungen betreffend den positiven Einfluss der Visuellen Entwicklungsmethoden wurden nicht erfüllt. Die Ergebnisse bestätigen, dass das Niveau des Einübens und des Verständnisses im erfolgreichen Unterricht sehr wichtig sind.