The Analysis of a Case Study for Group Programming Learning

Thais Helena Chaves de Castro[1,2], Hugo Fuks[1], Marcos André Fernandes Spósito[2], Alberto Nogueira de Castro Júnior[2]

[1]Informatics Department (PUC-Rio)
r. Mq. de São Vicente, 225 RDC – Gávea - 22453-900 – Rio de Janeiro – RJ – Brasil

[2]Computer Science Department (UFAM)
Av. Gal. Rodrigo Otávio Jordão Ramos, 3000 – 69077-300 – Manaus – AM – Brasil
{tcastro,hugo}@inf.puc-rio.br, {mafsposito,alberto}@dcc.ufam.edu.br

Abstract

This paper describes a case study on group programming learning that has as central components the identification of specific demands for such domain and the requirement analysis for collaborative programming learning, using the record of activities and codes developed.

1. Introduction

The computer science academic community has always sought to understand why programming is a difficult task and how the methodologies adopted in introductory courses influence the way in which students learn how to perform this task. With respect to learning, regardless of the area under consideration, it is well known that group work fosters in the individual the development of abilities and strategies for the solution of problems that are of extreme importance for the construction of knowledge in that domain [23].

Group work has proven to be a necessity both in the job market and in education. From an industrial standpoint, the increasing demand for products and services has caused heightened competition and has lead companies to search for convergence in their activities so as to avoid duplication of efforts for the same task, thus prioritising collaboration among the various teams [3]. From the point of view of education, when collaborating students can see their pairs' points of view and jointly build a solution for the problem presented. The effect of the contribution is easily perceived when following a course running in a Learning Management System, where one is able to analyse substantial differences in the refinement of registered solutions [1].

This article describes a case study conducted with first year students of the Computer Science and Computer Engineering undergraduate courses. The objective of this case study was to verify the adequateness and acceptability of group programming according to quantitative and qualitative analyses of the content developed by 9 student groups in the context of this work.

In Section 2 a bibliographical revision on programming learning is presented, emphasizing the difficulty in keeping a record of the cognitive processes involved. In section 3 the aforementioned case study followed by its quantitative and qualitative analyses are presented. Finally, in section 4 we present this work's conclusion.

2. The programming activity and its learning

Learning concepts and methods for the construction of computer programs is not trivial, as it requires the use of high level abilities and a high dose of abstract reasoning. In [8] it is emphasised that programming involves reasoning more than any other ability. But programming is also an engineering task, once it deals with the production of artefacts that must satisfy quality requirements and be subjected to verification.

In [19] it is pointed out that in introductory courses students seldom learn problem solving techniques. What normally happens is that students find great difficulty in applying their previous skills. This ends up becoming a source of fear and frustration, thus fostering evasion. In the work described in [7] an effort to develop a new programming course based on laboratory sessions, several activities were planned such as online discussions, programming exercises in pairs, reading of texts from the Internet, reflection annotations, diary entries and collaborations using the pair review process to criticise colleagues' replies to a given topic.

The latter article deals with the transformation from a methodology involving theoretical and practical classes to one that uses only practical classes, with distinct activities well distributed among the sessions. In spite of the preoccupation with developing questionnaires to create in the students the habit of reflection, such methodology proved inefficient to detect confusion in the grasping of concepts. As to the programming exercises carried out in pairs there is no evidence of an improvement in performance as a result of this technique given that there was no record of pair activities or even a control group.

Following the learning evaluation line, the work presented in [6] describes a combination of some evaluation techniques in an introductory computer course and demonstrates through statistical analyses the differences and relationships among these techniques. The course was planned following the premise that before learning how to program students must be able to solve problems. Therefore, firstly students solve problems without the use of a programming language and later learn how to use it to represent solutions.

After the initial phase of problem solving, the course continues with six laboratory sessions where students must individually solve problems, being allowed to consult their pairs whenever they deem necessary. It must be pointed out that problem complexity increases as the sessions advance. Once this phase is over a case study is proposed consisting in the development of a program (normally a game) by small groups (2 to 4 members). A learning evaluation is conducted statistically comparing student performance in the laboratory sessions, case study and controlled individual practices without consultation (such as tests) carried out twice throughout the course.

The contribution of the above mentioned work is the statistical analysis of correlations among the evaluation mechanisms used. Although it is important to know if the students are being evaluated by the most efficient methods for their learning, other factors are disregarded, as it occurs with the nature of the work developed in groups – for instance, there is no control of the development of each work, making it impossible to ascertain if a given task was performed by a single student, which would cause an error in the correlations.

Once again using evaluation models to teach programming, the work described in [15] uses the students pre-evaluation as a basis to categorise them in learning stages as defined in Bloom's taxonomy. From there the course is formatted in such way as to offer differentiated activities for students in the different stages of training.

In an effort to identify aspects that can facilitate programming learning, the work described in [9] states that through students' answers to questions such as "what is programming?" it is possible to define a presentation order for the programming paradigms. The authors depart from the principle that students need to know what programming learning really is in order to actually learn. The majority of the answers surveyed suggest that they must first be exposed to a more structured reasoning before being presented to object abstractions in OOP.

What is really necessary to facilitate programming learning is still an open question. Although the above-mentioned articles attempt to find an answer to this question, there are no works in the reviewed literature that have succeeded in establishing undisputed methods and techniques for group programming learning. On the other hand, there have been initiatives whose focus is to create and maintain student interest in the course using concepts of extreme programming, already widely used in development teams in the software industry.

3. A Case Study for an Introductory Programming Course

A case study on introduction to programming was carried out in the first semester of 2007 with two groups of students enrolled in the first semester of the Computer Science and Computer Engineering courses at the Federal University of Amazonas.

The objective of this case study was to propitiate students the experience of developing solutions for complex problems through the distribution of tasks, negotiation, composition of partial solutions and successive refinements. This was achieved working in groups of up to 5 members who were also responsible for recording the activities developed along the several work stages, using a version control environment denominated AAEP [1]. At the end of the course, as a final activity, a group task was posed. The solution had to be supplemented by the record of the interactions among team members. Then, the analysis and interpretation of these data based on classification, codification and tabulation processes took place.

Besides developing code and keeping the record of interactions the students also answered a questionnaire, which was submitted to a quantitative analysis. Finally, they spoke freely of their difficulties and of the work's development stages, being these comments subsequently submitted to a qualitative analysis.

3.1 Quantitative analysis

The aim of the questionnaire was to figure out the adherence level to the learning progression scheme proposed as a way to move from individual to group learning through the integration among team members. The data was analysed according to the absolute distribution suggested in [14] described in Table 1.

Table 1 - Distribution of the Experimental Study Criterion

Criterion	Answers		
	Totally	Partially	Not done / Not observed
Suggested sequences	7	2	0
Goals reached	7	2	0
Problem solved	7	1	1
Search for similar code	4	2	3
Satisfactory annotations	6	3	0
Integration among team members	4	3	2
Reuse of team's own code	8	1	0

As the analysis of Table 1 points out, there were 9 groups of respondents and in all criteria there was a predominance of "Totally" answers that indicates satisfaction and achievement in performing the group task. In most criteria, the majority of the answers fell in the first column (Totally) indicating that there was an attempt of following the specification of the problem. However, in the criteria "search for similar code" and "integration among group members" the answers are almost equally distributed among the three columns, indicating that the respondents were note quite comfortable on dealing with them.

3.2 Qualitative analysis

For the qualitative analysis the objects of the research were not reduced to the questionnaire's criteria; instead, they were studied entirely in their daily application context, the latter already indicated in the case study description. As for the qualitative research it must be pointed out that, according to [10], the reflections of the researchers regarding their actions and observations in the field, their impressions, feelings and so on become data in themselves and constitute a part of the interpretation.

The groups' and researchers' reflections were collected and turned into texts based on their development reports put together by the groups and the annotations of the observer/researcher. Data documentation is not simply a neutral recording of reality, but rather an essential stage of its construction in the process of qualitative research. The interpretation of data is geared either to the codification and categorization or to the analysis of sequential structures in the text.

Table 2 - Difficults felt by the groups

Informers	Description of difficulties felt by the groups
A	Interpreting the statement of some questions
B	Transforming the solution sketches into Haskell code; gathering the entire team; reaching consensus; joining all the solutions.
C	Understanding Haskell's syntax; joining all the solutions.
D	Understanding how to build the program; building the program adequately; verifying errors; using the Hugs interpreter
E	Refining the solutions
F	Implementing the code
G	Using Hugs' interpreter; understanding Haskell's syntax
H	Refining the solutions; Dealing with recursion
I	Interpreting the statement of some questions; planning the solution; understanding Haskell

Two aspects were taken into account in order to analyse the texts of the development reports: the difficulties felt by the groups and the successes reported in the conclusion. The methodological procedure described by [16], proposing three techniques: a) abbreviation of content analysis; b) explanatory content analysis, and c) structuring content analysis, was also used. In order to summarize these texts at a higher abstraction level, a reduction of the material (the texts of the reports) was achieved through the omission of statements as the first technique proposes. According to Table 2, the difficulties described above were found. Table 3 presents the conclusions reported by the groups in their development reports.

Table 3 - Conclusions supplied by the groups

Informers	Description of the conclusions supplied by the groups
A	It was very productive; we put our knowledge into practice; we learned how to work in a group
B	It taught us how to work as a team, helping each participant to mature and to learn how to deal with difficulties and each other's differences; there was a group commitment in all the performed activities
C	Not reported
D	Not reported
E	It demanded a group joint effort; it demanded connection and, above all, consensus among all group members; communication among team members was kept mainly through the Internet (Chat and e-mail). I
F	Not reported
G	Not reported
H	Not reported
I	A group joint discussion was necessary to find a viable solution

In order to clarify diffuse, ambiguous or contradictory texts involving content related to the application's context such as information regarding the author, generative situations, etc, the researchers used the explanatory content analysis technique. Based on this analysis, the researchers' perceptions of groups' difficulties are presented in Table 4.

Table 4 - Perceptions of the difficulties

Groups	Researchers' perceptions of the difficulties felt by the groups
A	Interpreting the statement of some questions
B	Implementing the code; gathering the entire team, reaching consensus, joining all the solutions.
C	Implementing the code; joining all the solutions.
D	Understanding how to build the program; implementing the code; verifying errors; using the Hugs interpreter
E	Refining the solutions
F	Implementing the code
G	Using the Hugs' interpreter; implementing the code
H	Refining the solutions; dealing with recursion
I	Documenting the activity developed; planning the solution; implementing code

Regarding the conclusions reported by the groups, it is possible to formulate the following explanatory paraphrase: It is necessary to work in group and there is a demand for commitment, effort and agreement from the participants. The proposed activities were beneficial and represented a good opportunity to put into practice everybody's acquired knowledge in programming.

The groups experienced several difficulties mostly related to the coding of the proposed solution in the Haskell language. Although solution planning was very difficult for the students for its outcome depends on their coordination and interaction, the main difficulty reported was related to skills in programming techniques and knowledge of the specific language. This entails that programming learning can be more efficient when conducted in a group following a model or scheme that facilitates this process.

Finally, the majority reported that the experience of carrying a programming activity in group was of great value. It is also observed that as important as group division was to perform the tasks, with the effective participation of each group member.

4. Conclusion

This article presents the analysis of a case study whose objective was to identify the difficulties in group programming learning, following the previous work described in [4,5]. As a reflection of the teacher researcher in this process, the performance of the groups was generally satisfactory. Comparing the implemented codes and their companion reports with the researcher's annotations it is possible to find a match, which often indicates difficulties in putting together the parts of the program developed individually in independent modules to build and refine the complete program.

The triangulation of data reported here, which according to [14] regards the use of different data sources and should not be mistaken as the use of different methods for the production of data, shows the same difficulties found in collaborative software development [24].

Finally, based on the first results of this research it is possible to propose a way of teaching in the introductory programming courses, which can provide a means to introduce collaboration into activities historically performed individually. In this direction, a programming learning scheme to group programming resulted from this work.

Acknowledge

This project is partially sponsored by the Brazilian Ministry of Science and Technology through the CTInfo project grant nº 550865/2007-1. Hugo Fuks is sponsored by CNPq individual grant nº 301917/2005-1 and also receives grant from the FAPERJ project "Cientistas do Nosso Estado" and it also receives resources from project ColabWeb – Proc.553329/2005-7, CNPq/CT-Amazônia n.27/2005.

References

[1] Almeida Neto, F. A. ; Castro, T. ; Castro, A. N. "Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação". In: XVII Simpósio Brasileiro de Informática na Educação, 2006, Brasília. Simpósio Brasileiro de Informática na Educação. Brasília: Gráfica e Editora Positiva Ltda, v. 17. p. 184-193. 2006

[2] Beck, K. "Extreme Programming Explained". Addison Wessley. Reading, MA, USA. 1999

[3] Brooks, F. "The Mythical Man-Month: Essays on Software Engineering. 2nd Edition. Addison-Wesley Publisher. ISBN 0-201-8359-9. 1995.

[4] Castro, T. H. C. ; Castro Jr, A. N. ; Menezes, C. S. ; Boeres, M. C. S. ; Rauber, M. C. P. V. "Utilizando Programação Funcional em Disciplinas Introdutórias da Computação". In: XXII Congresso da Sociedade Brasileira de Computação - X Workshop Sobre Educação em Computação, 2002, Florianópolis-SC. XXII Congresso da SBC. Porto Alegre : Sociedade Brasileira de Computação, 2002. v. 4. p. 157-168.

[5] Castro, T. ; Menezes, C. S. ; Castro, A. N. ; Oliveira, R. S. C. ; Boeres, M. C. S. "Enhancing Programming Understanding through Conceptual Schemas in Introductory Courses". In: CLEI Eletronic Journal, vol. 9, number 2, http://www.clei.cl/cleiej/volume.php. 2005

[6] Chamillard, A. T. and Braun, K. A. "Evaluating Programming Ability in an Introductory Computer Science Course". In: SIGCSE 3/00. ACM 1-58113-213-1/00/0003. Austin, Texas, USA. 2000

[7] Clancy, M.; Titterton, N.; Ryan, C.; Slotta, J. and Linn, M. "New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course". In: SIGCSE, ACM 1-58113-648-X/03/0002. 2003.

[8] Dijkstra, E. "On the Teaching of Programming, i.e. on the Teaching of Thinking". In: Selected Writings on Computing: A Personal Perspective. Springer-Verlag NY. 1982

[9] Eckerdal, A. and Berglund, A. "What Does It Take to Learn Programming Thinking?". In: ICER'05, ACM 1-59593-043-4/05/0010. Seattle, Washington, USA. 2005.

[10] Flick, U. "Uma introdução à pesquisa qualitativa". 2nd ed. Porto Alegre: Bookman, 311 p. 2004

[11] Goodhue, Dale L.; Thompson, Ronald L. "Task-technology fit and individual performance", MIS Quarterly, 19, 2, 213-236. 1995

[12] Herr, K. and Anderson, G.L. "The action research dissertation: A guide for students and faculty". Thousand Oaks, CA: Sage Pub. 2005

[13] Mayring, P. Qualitative Inhaltsanalyse. Grundlagen und Techniken. 7th edition. Weinheim: Deustcher Studien Verlag. 1997.

[14] Levin, J. "Estatística Aplicada a Ciências Humanas". 2nd. Edition. Editora Harbra Ltda. ISBN 85-294-0207-3. 1987.
[15] Lister, R. and Leaney, J. "Introductory Programming, Criterion-Referencing, and Bloom". In: SIGCSE, ACM 158113-648-X/03/0002. Reno, Nevada, USA. 2003.

[16] Mayring, P. "Quantitative Inhaltsanalyse. Grundlagen und Techniken. 7th edn. Weinhein Studien Verlag. 1983.

[17] McDowel, C.; Werner, L.; Bullock, H. and Fernald, J. "The Effects of Pair-Programming on Performance in an Introductory Programming Course". In: SIGCSE, ACM 1-58113-473-8/02/0002. Corvington, Kentucky, USA. 2002.

[18] McGrath, J. E., & Hollingshead, A. B. "Putting the "group" back in group support systems: Some theoretical issues about dynamic processes in groups with technological enhancements". In L. M. Jessup & J. S. Valacich (Eds.), Group support systems: New perspectives (pp. 78-96). New York: Macmillan. 1993

[19] Mckeown, J. e Farrell, T. "Why We Need to Develop Succcess in Introductory Programming Courses". In: CCSC – Central Plains Conference, Maryville, MO. 1999

[20] Mendes, A. J.; Gomes, A.; Esteves, M; Marcelino, M. J.; Bravo, C. and Redondo, M. A. "Using Simulation and Collaboration in CS1 and CS2". In: ITiCSE'05, ACM 1-59593-024-8/05/0006. Monte de Caparica, Portugal. 2005.

[21] Mendes, E.; Al-Fakhri, L. and Luxton-Reilly, A. "Investigating Pair-Programming in a 2nd-year Software Development and Design Computer Science Course". In:

ITiCSE, ACM 1-59593-024-8/05/0006. Monte de Caparica, Portugal. 2006.

[22] Mendes, E.; Al-Fakhri, L. and Luxton-Reilly, A. “A Replicated Experiment of Pair-Programming in a 2nd-year Software Development and Design Computer Science Course”. In: ITiCSE, ACM 1-59593-055-8/06/0006. Bologna, Italy. 2006.

[23] Sharan, S. (Ed). “Handbook of Cooperative Learning Methods”. Praeger Publishers. ISBN 0-275-96746-8. 1999.

[24] Walz, D., Elam, J. And Curtis, B. “Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration”. In: Communications of the ACM. ACM 0002-0782/93/1000-062. Vol. 36, no. 10. October, 1993.