FLESHING OUT CLUES ON GROUP PROGRAMMING LEARNING

Thais Castro[1,2], Hugo Fuks[1], Leonardo Santos[2] and Alberto Castro[2]
[1]*Department of Informatics, Pontifical Catholic University of Rio de Janeiro, r. Mq. de S. Vicente, Rio de Janeiro, Brazil*

[2]*Department of Computer Science, Federal University of Amazonas, av. Gal. R. O. J. Ramos, 3000, Manaus, Brazil*

Keywords: CSCL, Group programming learning, Programming in groups.

Abstract: This work examines the findings of a case study carried out in the first semester of 2008, which uses a programming progression learning scheme, from the individual to group programming. This approach implies the generation of conversation logs among students as they take part in group programming. Supporting strategies are the evidences fleshed out through those logs. These strategies will guide the teacher on his inferences in the next group programming practical sessions.

1 INTRODUCTION

The extremely competitive scenario in which organisations strive to survive nowadays has imposed profound changes on Human Resources in general and on individual and group abilities in particular. As a result, there is a clear demand for project-oriented, team-based, collaborative approaches to be explored at early stages during continuing education. In addition to this, pedagogical practices and teaching strategies need to evolve in order to suit those new requirements. Collaborative learning methods have also an important role in contributing for that challenge.

The use of internet based tools represents broadened opportunities for recording, organizing and reusing experiences on learning and working at both individual and group levels. In order to elicit knowledge on how to improve collaborative practices, it is necessary to analyse every experience on the artefacts produced.

In this context, this paper reports some aspects of a case study on group programming learning, focusing on the analysis of interaction dialogues, fleshing out clues on how students organize themselves in order to act under a collaborative scheme. By analysing, grouping and comparing students' interaction through a Learning Management System, it was possible to identify opportunities for intervention. This may result on new guidelines for the current supporting strategies.

2 CONTEXTUALIZATION

At UFAM, a Brazilian University, there is a fifteen year experience in introductory programming learning using the functional programming paradigm. The emphasis in the introductory course is in problem solving, as described in (Castro et al, 2002). Besides the functional approach, that research group conducted experiments using collaborative methods to represent problem solving knowledge (Mendonça et al, 2002) (Pereira et al, 2002) (Silva et al, 2002).

It is a common thought (Brooks, 1995) that there is a strong link between the software engineer performance and its academic background. Thus, in the introductory programming courses it is mandatory the use of techniques based on problem solving, given that problem solving is a necessary ability for a software engineer. That is also the reason why training for collaborating is an important part of undergraduate curricula.

In order to collaborate, initially students have to be aware of some problem solving strategies. Consequently, a pilot study was conducted in 2004 for probing which were the most difficult topics in the introductory course (Castro et al, 2005).

That pilot study was applied to two freshmen groups. In their weekly practical class, students got one or two exercises that were previously solved during the theoretical class. They could consult the Internet or any other bibliography and also discuss

Castro T., Fuks H., Santos L. and Castro A. (2009).
FLESHING OUT CLUES ON GROUP PROGRAMMING LEARNING.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Software Agents and Internet Computing*, pages 68-73
DOI: 10.5220/0001951700680073

the exercise with other students or trainees. From roughly 80 students who attended the course, 10 constituted the observation group. The pilot study follow-up was based on a qualitative analysis of their annotations, before and after they solved the exercises. After every laboratory session, the researchers read the students', trainees' and research assistants' annotations (5 assistants observed the observation group while they were solving the exercise). Then, annotations were triangulated in order to find out which students should be invited for an individual interview, based on the clinic method protocol, as proposed by Jean Piaget, explained in (Del Val, 2002).

During the aforementioned case study analysis the researchers recurrently stumbled on following the changes done to the code. In order to solve this tracking problem a tool, called AAEP, was developed for keeping track of the students' code evolution by the pairing of versions. Later on, for the purpose of classifying the code evolution in one of these three categories, namely, syntactic, semantic and refactoring another tool named AcKnow (Castro, Fuks, Castro, 2008) was developed.

However, when students work in groups it is difficult to know who is responsible for each code fragment and who is collaborating with whom. Besides that, according to the literature in Cognitive Science (Cohen, 1997), students learn more when they are working in groups and programming is definitively a cognitive activity (Weinberg, 1971). Thus the proposal of integrating AAEP-AcKnow into a groupware, that took place in both 2007 and 2008.

In the following section, literature about group programming is presented aiming to introduce group programming learning.

3 A CASE STUDY FOR GROUP PROGRAMMING LEARNING

A case study was carried out in the first semester of 2008 in order to evaluate students' engagement on a programming introductory course. This article deals with the conversation follow-up, one of 4 supporting strategies identified in the context of a research project. One supporting strategy was discussed in (Castro, Fuks, Castro, 2008) and the others will be discussed in future works, based mainly on (Gerosa et al, 2006) and (Pimentel et al, 2006). The case study was applied to 60 Computer Science freshmen students. Incrementally, exercises got more complex as collaboration becomes necessary, according to the programming progression learning scheme.

3.1 The "How-to-Do" Analysis

The case study was based on a progression scheme (Castro, Fuks, Castro, 2008b) that comprises nine sequential exercises together with their coding, companion reports and conversation logs. Two additional warming up exercises precede the ninth one, which resembles a programming marathon.

The How-To-Do analysis consists of inferring the communication purpose on each conversation turn. For instance, when a student says "Hey folks! I've done mine and I found out that it was simple", his objective is just to inform. In Table 1 there is a description of each message category found in the conversation log, followed by its instance. That gives the researcher an overall idea about group behaviour. Then, the message body was further analysed. Table 1 below shows the categories found.

Table 1: Utterance Categories.

Category	Instance
Making an artefact available	"My functions…"
Informing	"Guys, the problem ins't so difficult…"
Clarifying	"I couldn't log in before."
Confirming	"I've annotated it already…"
Asking	"Does someone want to add something else on the report?"
Suggesting	"…everyone should try to create a solution to each question on his own way…"
Calling attention	"Hey, Guys! Let's do the exercise!"
Pointing a mistake	"I think you made a mistake when you defined the output as the type int…"
Explaining	"…What I did was to use the 2nd question that…"

In phases 1 and 2, students start to learn how to collaborate using a chat tool. The whole class took part in the discussion, but coding was an individual activity.

Next, in phase 3, coding was still an individual activity, but then they had to choose the best code among all the participants' and also justify why that code was the chosen one. Almost all the groups generated long conversation logs and the discussions run deep in eliciting the requirements for the best code candidates. These conversations demonstrate that students are quite comfortable in testing each

other's code and comparing them with the course's notion of efficiency and efficacy.

In phase 4 they distribute the pre-divided parts of the exercise to all group members and then discuss it in order to prepare their wiki-based group report (that comprises the individual coding and the combined group coding together with the suggested problem solving method). Their forum-based conversation logs show at this point that although some groups could collaborate in a rudimentary way they lacked the ability to solve the problem as a group, passing the responsibility to combine the parts to one self-appointed group member. Almost half of the groups just made available their individual members' coding. Their discussion log showed that group members had no understanding about the exercise as a whole and that one student combined all the individual codes, making the necessary adjustments. A few groups discussed the task and really solved the exercise together, suggesting changes in each other's codes, where it was the case. One group did not use the forum tool to discuss, justifying that they had face-to-face discussion. Only one group did not do anything at all.

Phase 5 asks the groups for splitting the exercise into functional parts for distribution among group members. That entailed heated discussions showing that they moved to a next level in their collaboration. One possible reason for this improvement is that the groups that had difficulties or misunderstandings about how to conduct themselves in a collaborative way got feedback pointing out where in the problem solving process they were not able to collaborate. Another possible reason was that students got progressively more involved on their activities as they kept carrying on the exercises posed within the progression scheme.

Phase 6 of the progression scheme is the programming marathon, where they work in small groups of three members. In this phase, group formation is left for the students to form new groups of their likings. Surprisingly, not that many new groups were formed. During the four hour collocated marathon, record keeping was not mandatory, although they had to write the wiki-based report.

Summing up, it could be observed that as students progressively solved more exercises, they produced better codes. Interestingly, students resorted to collaboration whenever they did not know how to solve the entire exercise a priori.

3.1.1 A Closer Look at Exercise 5

The conversation logs commented below were translated from Portuguese. For that reason, some misspellings and/or inappropriate use of English are still in the text to give the reader a more accurate view of the conversation.

Exercise 5 (in Phase 4) stand as a bottleneck because, for the first time during the course, students are urged to think about the whole problem and its solution collectively. During the problem solving process they have to accommodate each other's views and negotiate when there is a conflict of understanding. For that reason, this subsection presents exercise 5 in order to flesh out clues on group programming learning within each group's conversation log.

Table 2 describes exercise 5, noticing that the teacher proposes a specific method for solving it. Some groups adapted in order to solve the exercise according to their own characteristics.

Table 2: Exercise 5: Doctors allocation problem.

In a clinic, as soon as a patient arrives at the hospital, she receives an attending number. There are always three available doctors per shift, and they will receive incoming patients depending on the number of patients a doctor already has in her to attend list. The doctor who has fewer patients in her list gets the next one. Using tuples, we can define the following input: available_doctors (("dr. A", 4, 23), ("dr. B", 1, 13), ("dr. C", 3, 27)), where the 2nd term of each tuple refers to the number of patients in that doctor's list and the 3rd term refers to the last patient attended by that doctor. Based on this input, write a script in Haskell that, for a given incoming patient, choose in which doctor's to attend list she should be allocated to. …

Group 1 does not discuss about either the problem understanding or the problem solving, including the division of work and the process of joining all the individual solutions together to build a unique group solution. Contrary to that, each group member makes available his solution.

One member of group 2 in a self-appointed way leads the group. Another one assumes the division of work. The discussion is rudimentary about the general understanding, although it gains strength when the matter is the work itself. The last topics in the conversation log deals with suggestions. The leader examines all codes and points out what could be improved in a specific one, as shown in fragment below (Table 3).

Table 3: Suggestions.

StD1:	StK1 HAS TO CORRECT THE FUNCTION AND TO REDO POLYA'S STEPS, StD2 AND StF1 HAVE TO CORRECT POLYA'S STEPS. StD2, I've already written above what is necessary in your case. StF1, the inputs are a list of tuples, even when the function you're working with uses the previous function's result. I've corrected there in Polya's steps. E.g.: Inputs: [("dr. A", 4, 23), ("dr. B", 1, 13), ("dr. C", 3, 27)]. Outputs "dr. B" StK1, your function is wrong, it has to return the complete list of tuples, as the doctor's tuple required in the input will have the 2nd term -1 and the last one will be the number of the patient added to the input. Examples: Inputs: "dr. A" 28 [("dr. A", 4, 23), ("dr. B", 1, 13), ("dr. C", 3, 27)] Outputs: [("dr. A", 3, 28), ("dr. B", 1, 13), ("dr. C", 3, 27)]

Group 3 follows the same pattern: one student self-appointed as group leader and division of work. There is a general question about understanding the exercise that the leader emails to the teacher. One student makes available his code and report which apparently makes another student uncomfortable. Immediately she poses a question, starting a clarifying discussion, as shown in the fragment below. Unfortunately, the group report was generated face-to-face and there is no discussion log about it, only the wiki indicating that some group members edited it.

Group 4 differs because more than one member is leading the group. One student poses a topic about some peculiarity found in the exercise. Based on that, another one discusses the nature of the exercise and a third one suggests a slightly different problem solving method from the one proposed by the teacher. Everybody follows the third member's suggestion triggering discussions about each member's solution. The fragment below (Table 5) shows when a student figures out the nature of the exercise and another one proposes a way to do it and starts leading the process.

Group 5 makes the codes available straight away and this apparently is a result of a just finished face-to-face conversation.

Group 6 does not discuss the exercise as a whole, keeping the conversation restricted to two members and only about the last function.

Table 4: Clarifying discussion.

StV1	Hey folks! I've done mine and I found out that it was simple. aux_menores x xs = [y | y <- xs , y < x] indice_menor xs = [i | i <- [0..length xs-1], aux_menores (xs!!i) xs == []] But even if it was simple, I'd like you to look at the end of the function "indice_menor xs" (aux_menores (xs!!i) xs == []), 'cause it was there where I had more difficulties. I was trying to do two extra auxiliary functions, where one of the extra functions is aux_menores and the other, which I didn't make available, indicates the lowest number. So, in the main function could be indicated "xs!!i==número menor". But I realized that this could be done directly, with only one extra function. Anyway, I'd like you to test it!!
StF1	Well, mine was pretty short, I thought it was odd, but I think it is complete, as it was a simple question. What I did was to use the 2nd question that shows the lowest indices and use them to show the doctor with fewer patients. It follows: medicos_menos_pacientes = disp!! indice_menor
StV1	Man, explain that in details…. Because… I can't find a method that fits my solution. uhh.. Did you test it? Because, mine is "indice_menor xs".

Table 5: Working on a solution.

StJ1	I think the problem is sequential… Each one of the required functions is easy if we use the previous one, once one depends on the other… I believe the best way of dealing with the exercise is doing it sequentially each function reusing the previous functions, creating extra ones, when it is the case.
…	…
StR1	As far as I can see everybody in the group agrees about the fact that the problem is sequential and consequently reuses every item in the next. So, because there is not much time left, ideally everyone should try to create a solution to each question on his own way, and separate the best ones, to combine them in order to write one group solution. In case of wasting too much time in one question, try to think about the next, continuing the approach started by other member or even modifying it…

Group 7 makes use of the most common way of solving the exercise. One student leads the discussion, asking for the division of work and at the same time makes available his code. Another student

subdivides the work and asks everyone to make their solution available as soon as possible, together with their respective explanation. All members comply. One member of the group reads all the codes and finds out a mistake and its probable cause (shown in the fragment below). Something that deserves attention is at the end of the discussion, when one student says that he has not done his part yet, but needs no help. In the very end of the discussion he makes his solution available. The fragment below, Table 6, shows how the student asks for help

Table 6: A students asks for help.

StP1	Oops! Sorry guys. I haven't tested it yet and only now I saw it has an error. But I know the reason why. It's because StD3 function came from a sequence of functions of the type "one fosters the other" and that's why I corrected it by copying and pasting every function from the first to the third one.

Initially, group 8 discusses about general questions related to the exercise, sharing some expertise. One member complies with another member's suggestion and everyone continues reusing the functions just done. There are still more questions about the exercise, but the group remains collaborative, except for one member, who does not take part in the conversation and does his part without reusing or revising the functions that have been done already (this is shown in Table 7).

Table 7: Working alone.

StF2	I've done the 5th question. As I didn't used the questions which have been done before it is big, but, basically it has the same functions that have been done until the 4th one.

Group 9 behaves similar to group 1, but with absolutely no discussion. Everyone makes his codes available and one member is responsible for the tests and group solution.

Group 10 does not have any discussion log and did not do the exercise at all.

The way each group behaves on their first group exercise fleshes out clues about the cause of possible difficulties in collaboration and evidences of successful use of an informal collaboration method.

In the aforementioned description of the way each group collaborates in the exercise 5, most of the groups tried to collaborate and behave as a group, but they did not achieve the whole meaning of collaboration. The recording of these group exercises were very important, because it made easy for the teacher to identify these groups' difficulties and preparing them for the next exercise.

Other groups, namely groups 2 and 4, properly performed the work, without any difficulties. Group 4 even used a new problem solving method, which was beyond the expectation at this point. Only two groups (1 and 9) completely ignored the fact they were doing their exercise as a group and group 10 did not started to talk about the exercise.

4 CONCLUSIONS

Programming learning, as other intellectual activities, can benefit from collaboration. However, group programming learning also has to tackle several difficulties, common when developing abilities related to an effective participation in a group.

In this work, we have shown how an Internet based, record-oriented, collaborative and progressive scheme can be used to allow both: knowledge elicitation on the development process of collaboration ability within the group; and identification of opportunities for instructional intervention.

The conversation logs analysis produced an insightful view of a collaborative learning approach used in an intricate domain, expressed in the findings:

- Teacher indication of a specific way of solving an exercise is important. As groups acquire more expertise, they feel more confident to try different approaches;
- Self-assignment is the most common criteria for initial designation of a group leader. More important though, is that a discussion about the general understanding of the problem and task division is the subsequent step;
- Request for selection of an individual solution to represent the whole group is a good way of starting objective discussions and positive criticism over the artefacts;
- As students move forwards through stages of the progressive scheme, and problems get more complex, individual behaviour tends to be more responsible;
- Inspection of dialogues and correspondent solutions at each stage made possible for teachers:
    - identifying groups' specific difficulties and acting on that;

- having a more levelled set of groups at the next stage of the progressive scheme.

We are now using these findings as requirements for a knowledge-based framework to task guidance on supporting the teaching and learning process, using the collaborative setting presented.

ACKNOWLEDGEMENTS

Hugo Fuks receives individual grant from CNPq and FAPERJ. This research is financed by ColabWeb – Proc.553329/2005-7, CNPq/CT-Amazônia n.27/2005 and GroupwareMining – Proc.575553/2008-1, CNPq/CT-Amazônia n.055/2008.

REFERENCES

Brooks, F. 1995. The Mythical Man-Month (aniversary edition). Addison-Wesley Longman Publishing Co.

Castro, Thais H.C.; Castro Jr, Alberto N.; Oliveira, Rosane S.C.; Boeres, Maria C.S.; Menezes, Crediné S. 2005. Enhancing Programming Understanding through Conceptual Schemas in Introductory Courses. *CLEI Electronic Journal. Vol. 8, Num. 2, Pap. 4*. (http://www.clei.cl/cleiej/).

Castro, T., Fuks, H., Castro, A. 2008. Detecting Code Evolution in Programming Learning. In *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence*. Series: Lecture Notes in Computer Science, Vol. 5249. Sublibrary: Lecture Notes in Artificial Intelligence, pp.145-156.

Castro, T., Fuks, H., Castro, A. 2008b. Programming in Groups: a Progression Learning Scheme from the Individual to the Group. *FIE - Proc. of the 38th Annual Frontiers in Education Conference*. IEEE Catalog Number: pp F1F15-F1F20.

Cohen, S. 1997. What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite. In *Journal of Management*, Vol. 23, No. 3, 239-290.

Delval, J. 2002. Introdução à Prática do Método Clínico: descobrindo o pensamento das crianças. ARTMED Press.

Freudenberg, S., Romero, P., du Boulay, B. 2007. 'talking the talk': Is intermediate-level conversation the key to the pair programming success story? In *Proceedings of Agile 2007*, IEEE Computer Society, pages 84-91.

Gerosa, M.A., Pimentel, M., Fuks, H. and Lucena, C.J.P. 2006. Development of Groupware based on the 3C Collaboration Model and Component Technology. *12th International Workshop on Groupware – CRIWG* Lecture Notes on Computer Science LNCS 4154, Springer-Verlag, pp. 302-309.

McDowell, C., Werner, L., Bullock, H., Fernald, J. 2004. The Impact of Pair Programming on Student Performance, Perception and Persistence. In *International Conference on Software Engineering*, pp. 602.

Mendonça, A. P. ; Castro, A. N. ; Mota, E.S. ; Silva, L. S; Pereira, V. L. S. 2002. Uma Experiência com o uso de Mapas Conceituais para Apoiar o Método da Controvérsia Acadêmica. In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola, Florianópolis-SC-BR*. SBC Press, v. 5. p. 99-107.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S. 2003. Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*., pp. 359 – 362.

Pastel, R. 2006. Student assessment of group laboratories in a data structures course. In *Journal of Computing Sciences in Colleges*, v. 22, issue 1, pp. 221 – 230.

Pereira, V. L. S. ; Castro, A. N. ; Mendonça, A. P. ; Silva, L. S. 2002. Análise do método Jigsaw de aprendizagem cooperativa através da utilização de mapas conceituais. In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola, Florianópolis-SC. XXII Congresso da SBC*. SBC Press, v. 5. p. 181-188.

Peres, F., Meira, L. 2003. Educational software evaluation centered on dialogue: interface, collaboration and scientific concepts. In *Proceedings of the Latin American conference on Human-computer interaction.* Pp. 97 – 106.

Pimentel, M., Escovedo, T., Fuks, H. and Lucena, C.J.P. 2006. Investigating the assessment of learners' participation in asynchronous conference of an online course. *22nd ICDE - World Conference on Distance Education*. Publisher: ABED, Rio de Janeiro, Brazil, Sep, 3-6.

Silva, L. S; Castro, A. N. ; Mendonça, A. P. ; Pereira, V. L. S. 2002. Mapas Conceituais como suporte à estratégia de Investigação em Grupo: Uma experiência na Universidade. In: *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola. Florianópolis-SC. XXII Congresso da SBC*. SBC Press, v. 5. p. 163-172.

Stahl, G. 2006. Supporting group cognition in an online math community: a cognitive tool for small-group referencing in text chat. In *Journal of Educational Computing Reasearch*.

Weinberg, G. 1971. The Psychology of Computer Programming. Computer Science Series. Litton Educational Publishing.