Confirming the influence of educational background in pair-design knowledge through experiments

Gerardo Canfora°, Aniello Cimitile°, Felix Garcia*, Mario Piattini*, Corrado Aaron Visaggio°
°RCOST-Research Centre on Software Technology University of Sannio
Palazzo ex-Poste, viale Traiano, 1
82100 Benevento, Italy
{canfora, cimitile,visaggio}@unisannio.it

*Alarcos Research Group, University of Castilla-La-Mancha
Paseo de la Universidad, 4
13071 Ciudad Real, Spain
{Felix.Garcia, Mario.Piattini}@uclm.es

ABSTRACT
The sharing of tacit knowledge is a strategic factor for the success of software process, from a number of perspectives: training, project assimilation, and reducing noise in knowledge transfer. Pair programming is supposed to be a practice suitable for this purpose. Unfortunately, the building of tacit knowledge is determined by factors that are difficult to isolate and capture because they concern personal attitude and capability. Thus, we have focused on the possible causes forming the individual ability, that can be isolated and studied, such as the individual education background. We have applied the practice of working in pairs to the design phase. We have made an experiment and a replica in academic environment, in order to understand the relationship between the building of knowledge through the practice and the individual background. In this paper we discuss the replica and compare the results with the first experiment’s ones.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Management – life cycle, programming teams, software process models.

Keywords
Agile development, pair programming, knowledge management.

1. INTRODUCTION
Governing the dynamics of knowledge building and sharing is a wide recognized strategic key factor for business success.

Two approaches are usually adopted when dealing with knowledge management. The first one refers to the notion of knowledge as object, to be created, modified, stored, and re-used. It addresses usually the explicit knowledge, that can be codified in rules, data, formulas, and procedures, and transferred by using diverse kinds of documentation. The second one considers knowledge in a cognitive dimension, consisting of beliefs, ideas, values, schemata, and mental models. It refers to tacit knowledge, that is personal experience and capability used to solve problems and, for its intrinsic nature, it is hard to codify and consequently to transfer, too.

So far, Software Engineering concentrated efforts in defining tools and processes to optimize formalization, elaboration and sharing of explicit knowledge. Melnik and Maurer [12] identified some concerns and limits related to such an approach for knowledge management. Documentation is not easy to produce and maintain and has a low communication bandwidth. They conclude that “face-to-face channels offer the prospect of richer communication because of the ability to transmit multiple cues[…] important when there are high levels of equivocality (ambiguity) and uncertainty.” Communication and collaboration are key features of pair programming. Pair programming [16] is a practice of extreme programming [2], where two programmers, working side by side, develop the same piece of code. One programmer, usually named ‘driver’, actively writes the code while the other programmer, usually named ‘observer’, identifies tactical and strategic defects and issues. The roles are periodically switched.

A claimed benefit of pair programming is that it fosters knowledge leveraging between the two programmers, particularly tacit knowledge. This is confirmed by Cockburn [7]: the most effective form of communication is two people discussing at a whiteboard, that is two people sharing the same physical space and instruments.

Williams [19] found as an initial exploration that pair programming can help to reduce the effect pointed out in the Brook’s law [3]. It asserts “adding manpower to a late software project makes it later”, because it increases training costs, assimilation time (time lost learning about the project), and intercommunication overhead (work required to communicate,
formally or informally, among the team members). Pair programming allows new team members to learn more about the overall project.

Standing such considerations, and because system designing relies on tacit knowledge, the practice of working in pairs could facilitate the process of building design knowledge when contrasted with working as singletons. Similarly to pair programming, we name “pair designing” the practice where two designers work side by side at the same design document; one of the two actively edits the document whereas the second performs continuous review.

In this paper we present the replica of an experiment, made at the University of Sannio in Italy, aiming at understanding the relationship between pair designing and knowledge building. The replica was realized at the University of Castilla-La-Mancha in Spain, with the purpose of confirming the findings of the previous experiment.

The building of tacit knowledge through the practice is affected by a huge amount of factors difficult to capture and measure, such as personal attitude, capability, experience. We focus on one of these factors: the individual education background, in terms of classes of graduation.

The overall research goal is to analyze a design task with the purpose of evaluating how individual background affects the knowledge built with pair designing from the viewpoint of the designer, in the context of an actual student project.

The paper proceeds as follows: section 2 presents the related work; section 3 discusses the first experiment; section 4 shows the replica design and section 5 discusses the results; finally, section 6 draws the conclusions.

2. RELATED WORK

The current work is part of a family of experiment, aiming at evaluating the relationship between the practice of working in pairs applied to any phase of software process and the knowledge building about the ‘big picture’ of the system. The results of the first experiment about the relationship between pair designing and knowledge leveraging were discussed in [4]. Two main outcomes were obtained. First, along all the experiment subjects who worked in pair showed a greater knowledge with respect to those who worked as singletons. Second, the knowledge building was more stable for pairs than for singletons: the knowledge growth of pairs produce better programs. These results are enforced by some experiences in using pair programming coupled with individual written reports in introductory computer science courses. This research is reported in [14] and basically explores the perception of pair programming from student perspective as beneficial to their learning. Students perceived pair programming as valuable to their learning. Forsythe et al. [8], describe the development and preliminary validation of tools to assess tacit knowledge for leadership at three organizational levels of US Army. This work provides good suggestions on how capturing tacit knowledge for experimentation. The main concerns emerging from the state of the art are two: first, the greatest part of the study involves students rather than professionals; second, the analyses accomplished are mainly qualitative instead of quantitative. To our best knowledge, however, no study has been published that addresses pair designing.

3. THE FIRST EXPERIMENT: DESIGN AND RESULTS

In this section the first experiment is summarized in order to understand the context of the replica. The experiment was executed with the purpose of testing the following null hypothesis:

\[ H_0: \] the difference in education between the pair’s components does not affect the building of system knowledge.

The alternative hypothesis is:

\[ H_1: \] the difference in education between the pair’s components affects the building of system knowledge.

Subjects. The experiment was executed with students of the Master of Technologies of Software (MUTS) and Master of Management and Technologies of Software (MUTEGS), high education university courses for post-graduates, at University of Sannio (http://www.ing.unisannio.it/master/). Students of MUTS own a scientific graduation (engineering, mathematics, physics), whereas students of MUTEGS own an economic/humanistic graduation (economics, philosophy, literature, philosophy).

Both the courses provide the same basic education in computer engineering (operating systems, programming languages, network, database, and software engineering), but MUTS
students are specialised for developing and maintaining Software Systems, whereas MUTEGS students are trained for dealing with the economic and organizational issues of software lifecycle. The two Master courses are held contemporarily with the same professors, in the same campus and both last one year, during which the students follow theoretical classes and lab sessions, develop a large and complex project in connection with an enterprise, participate to seminars from international experts, perform a three month stage in software companies.

The subjects were organized as follows:
- 4 pairs with one MUTF student and one MUTEGS student;
- 5 pairs with two MUTS students;
- 5 pairs with two MUTEGS students;
- the other 16 subjects, MUTS and MUTEGS, worked as solo designers.

All the groups were formed randomly.

Variables. System’s knowledge represented the dependent variable and it was evaluated by grading a questionnaire, answered by subjects after having performed the maintenance tasks on the system. The questionnaires were evaluated in this way: each correct answer was evaluated 1; each incorrect answer was evaluated 0. The independent variables were the kinds of pairs by classes of subject’s graduation: scientific with scientific, non-scientific with non-scientific, and scientific with non-scientific.

Rationale for the sampling from the population. Students of Software Engineering course are suitable for such an experiment because they study software architecture and software system design. Furthermore they usually are employed as software architects or designers after the graduation. MUTS and MUTEGS students are a fine population’s sample, considered that they experienced an actual project work during the overall master. Since the students have comparable curricula, there is not a relevant bias in the sample.

Assignment. In order to evaluate the knowledge built by doing while working on the system, the assignment for the subjects consisted of improving the design of the system. The design of the system was formalized in UML and included: textual specification of the system’s requirements, two use cases diagrams, and two class diagrams (for a total of 15 classes). The design was developed by experimenters. Considered the time available, we preferred to avoid bulky documentation. The maintenance tasks were basically two:
- reduce complexity, by erasing entities or relationships between entities not fundamental for understanding;
- improve readability, by changing existing entities (use cases, actors, classes, methods), or adding new ones.

This kind of assignment was targeted at maximizing the knowledge built by doing; as matter of fact, maintenance needs the programmer to analyze in depth the system. The system design was realized by taking into account the knowledge of subjects, with the aim of making objects representative of the population.

The Process. The process of the experimental run was the following:
- each subject studied documentation for 30 minutes, individually;
- each subject answered an entry questionnaire, individually, for about 15 minutes. The entry questionnaire was aimed at establishing the baseline, i.e. level of knowledge of the system before working on it;
- the pairs and the solo designers performed the maintenance tasks for 2 hours;
- each subject answered an exit questionnaire individually, in order to understand the knowledge built by practicing the maintenance in the two different ways, pair and solo.

Questionnaires. We prepared two questionnaires, QA and QB, in order to measure the dependent variable. Both the questionnaires were distributed as entry and exit questionnaire, so that each subject had randomly QA (or QB) at entry and, conversely, QB (or QA) at exit. This avoided that the results depended on the questionnaire itself. The questions concerned architectural and functional aspects of the system. Although we would have liked to use a CASE tool, such as Rational Rose, or ArgoUML, we finally decided to use only pen and paper. The reason was that some subjects could be more familiar with this kind of tools and this could inject bias in the results. As consequence, we would have needed more time for preparation in order to equalize the ability of subjects to work with tools. We analyze only the exit questionnaire for evaluating the knowledge built. The entry ones helped only to assess the average level of system knowledge. In Table 1 the experimental design is provided, and in Table 2 the outcome of statistical tests are reported.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Treatment</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 MUTS</td>
<td>Paired MUTS MUTEGS</td>
<td>Requirement Specification; Use case diagram; Class Diagram; Entry questionnaire QA (or QB);</td>
<td>Modifications to Use Case Diagram and Class Diagram; Answered entry questionnaire QA (or QB); Answered exit questionnaire QB (or QA);</td>
</tr>
<tr>
<td>4 MUTEGS</td>
<td>Paired MUTS MUTEGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 MUTS</td>
<td>Paired MUTS MUTEGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 MUTEGS</td>
<td>Paired MUTS MUTEGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 MUTS</td>
<td>Paired MUTS MUTEGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 MUTS</td>
<td>Solo</td>
<td>Exit questionnaire QA (or QB);</td>
<td></td>
</tr>
<tr>
<td>8 MUTEGS</td>
<td>Solo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Outcome of the first experiment

<table>
<thead>
<tr>
<th>Tests</th>
<th>Rank Sum (a)</th>
<th>Rank Sum (b)</th>
<th>p-level</th>
<th>p-level 1 tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUTS-MUTS (a)</td>
<td>121,00</td>
<td>50,00</td>
<td>0,020</td>
<td>0,020</td>
</tr>
<tr>
<td>MUTS-MUTS (b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUTF-MUTS (a)</td>
<td>135,00</td>
<td>75,00</td>
<td>0,023</td>
<td>-</td>
</tr>
<tr>
<td>MUTF-MUTS (b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUTF-MUTS (a)</td>
<td>135,00</td>
<td>75,00</td>
<td>0,020</td>
<td>0,023</td>
</tr>
<tr>
<td>MUTF-MUTS (b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUTF (a)</td>
<td>54,50</td>
<td>116,50</td>
<td>0,049</td>
<td>0,054</td>
</tr>
<tr>
<td>MUTF (b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUTF (a)</td>
<td>57,50</td>
<td>78,50</td>
<td>0,270</td>
<td>0,278</td>
</tr>
<tr>
<td>MUTF (b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questionnaire A</td>
<td>540,00</td>
<td>406,00</td>
<td>0,161</td>
<td>0,179</td>
</tr>
<tr>
<td>Questionnaire B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following conclusions have been drawn: forming pairs with individuals with the same educational background (either
scientific with scientific or non-scientific with non-scientific emphasizes the expected benefits of pair designing. Coupling a person with a scientific background and one with a non-scientific background does not seem to improve the performance of the latter but to make worst the former.

4. THE REPLICA: EXPERIMENTAL DESIGN

This section discusses the replica. The replica varied the experimental subjects, according to the classification in [1] and was executed testing the same null hypothesis of the first experiment.

Subjects. The subjects were students enrolled at the Department of Computer Science at the University of Castilla-La Mancha in Spain. The first group was composed of 42 students enrolled in the final-year (third) of the Computer Science (BSc) in the specialization of Management; the second group was composed of 39 students enrolled in the final-year (third) of the Computer Science (BSc) in the specialization of System; and finally the third group consisted of 12 students enrolled in the final-year of the fifth year of MSc. The curriculum of the Spanish students and that of the MUTS students show many subjects in common; thus Spanish students can be assimilated to the MUTS students. The replica had a different experimental design with respect to the previous experiment: 32 randomized pairs where formed and 32 subjects were left working as solo designers.

<table>
<thead>
<tr>
<th>Table 3. Experimental design of replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
</tr>
<tr>
<td>64 students</td>
</tr>
<tr>
<td>32 students</td>
</tr>
</tbody>
</table>

Dependent and independent variable, the process, the assignment and the questionnaires remained the same but: the overall process in Spain lasted 2 hours and the experimental object was properly translated in Spanish by the native Spanish language speaking authors.

5. DISCUSSION OF RESULTS

In Table 4 the results of statistical tests are reported. We used Mann Whitney test because the data of samples did not show normal distribution and fixed the p-level at 0.05.

From the statistical tests the following outcomes can be derived:

- The null hypothesis cannot be rejected only when comparing Spanish pairs and Italian pairs formed by MUTS students.
- The difference between the results of the questionnaires is not significant, thus they do not affect the outcomes of the experiment.

Recall that the Spanish education background is comparable with that of the Italian MUTS students.

<table>
<thead>
<tr>
<th>Table 4. Statistical tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairs under comparison</td>
</tr>
<tr>
<td>Italy MUTS-MUTS(a)-Spain(b)</td>
</tr>
<tr>
<td>Italy MUTS MUTEFS(a)-Spain(b)</td>
</tr>
<tr>
<td>Italy MUTEFS MUTEFS(a)-Spain(b)</td>
</tr>
<tr>
<td>Questionnaire QA (a) Questionnaire QB (b)</td>
</tr>
</tbody>
</table>

The second and the third row confirm the findings of the experiment run in Italy: the 'background of pair' s components affects the knowledge built. Precisely, the closer the curriculum of both the designers is to scientific profile (like MUTS and Spanish students) the greater knowledge the designers build. The data in the first row reinforces this conclusion: as matter of fact, by comparing two groups of pairs, both formed by designers with scientific background, the difference in results is not statistically significant. Table 5 shows the descriptive statistics of data samples.

<table>
<thead>
<tr>
<th>Table 5. Descriptive Statistics of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairs</td>
</tr>
<tr>
<td>Italian MUTS MUTEFS</td>
</tr>
<tr>
<td>Italian MUTS MUTEFS</td>
</tr>
<tr>
<td>Italian MUTEFS MUTEFS</td>
</tr>
<tr>
<td>Spanish</td>
</tr>
</tbody>
</table>
Precisely, the best results are obtained when the background is scientific. We have detected some limitations in our experiment, to be overcome with future replicas.

The first limitation stands mainly in the experimental objects. The subjects used as the modeling language UML and design techniques traditionally thought in BSc courses. It should be interesting to evaluate learning of junior designers while faced with concerns of real world projects, including quality procedures, special architectural styles, and frameworks for complex systems, such as distributed, mobile, and services-based applications.

The second limitation is the questionnaire as a means to capture the tacit knowledge level and, thus, the knowledge building. Tacit knowledge for its intrinsic nature is hard to capture completely, and it cannot be formalized. Evaluating it by using questionnaires about the design of the system and design technique is an acceptable approximation. More sophisticated instruments could be more effective, but in this case the experimentation has to become multidisciplinary; it should include psychology research, at least.

The third limitation regards the construct. For sake of precision we must point out that the experiment focuses on student learning, rather than on comprehension-in-practice. With comprehension in practice we mean the knowledge that practitioners build by performing their daily work. Both are forms of tacit knowledge building, and both are intended to occur when people, either students or practitioners, perform development activities. However, at a finer analysis, differences can be identified. Students are committed in learning and whenever they work during classes or labs, also during the experiment, their efforts are headed in this direction. Practitioners do not necessarily concentrate their efforts in learning during their daily work. Comprehension in practice is a kind of unconscious learning and it occurs spontaneously when people face new problems. In this case, practitioners convey their efforts just on work: learning comes as a side-effect. Finally, in both the cases we can observe how knowledge is built by doing, but we must consider that a difference exists. The aim of students is to apprehend: the learning is the primary goal; on the contrary, the only aim of practitioners is realizing successfully their tasks: the learning, when occurs, is not a primary and voluntarily pursued goal.

These three limitations should be mitigated by considering the context of the experiment, that is student learning: this saves the consistence of the overall experimental structure, considering that the limitations are part of the experimental ‘vitro’.

### 5.1 Validity Threats

Four major validity threats are discussed in the following.

**Random heterogeneity of subjects.** Subjects come from three different undergraduate courses. This threat is minimal, because all of them owned the necessary knowledge to perform the tasks.

**Mono-operation bias.** We used one system design documentation as the only experimental object. Actually, we blocked this factor, that is the system the practice is applied to.

**Evaluation apprehension.** Questionnaires were not anonymous; this is an unavoidable threat, because professors were interested in knowing the students taking part to the experiment. Moreover, the students were volunteer.
Interaction of setting and treatment. The setting and the material were partial representative of real world. The design system was trivial in size, if compared with real system whose architecture is more complicated. But it should be considered part of the experimental environment: the subjects were students instead of practitioners.

6. CONCLUSIONS

A growing interest in the software engineering community is turning toward agile practices, such as pair programming. More precisely, a lack of experimental validation of supposed benefits constitutes one of the main concern.

Pair programming is expected to foster knowledge leveraging between pair’s components. We have applied pair programming to the design phase, naming it pair designing. We have conducted an experiment at University of Sannio, Italy and a replica at University of Castilla-La-Mancha, Spain in order to confirm the findings of the first experiment. This could help for using pair programming as a means for quickly training people and assimilating them in projects.

We found confirmations that forming pairs with individuals with the same educational background emphasizes the expected benefits of pair designing. Coupling a person with a scientific background and one with a non-scientific background does not seem to improve the latter but to make worst the former. This finding could be very useful in universities daily practice. In fact, in Europe due to the recent European agreements, university education is experiencing a turn from a teaching style to a learning style, in which students must learn by themselves with the guidance on teachers. When planning this kind of education it would be very interesting—in scientific subjects—to consider pair building in order to increase learning.

The experiment owns an (apparent) point of weakness: it is not executed in industrial setting. Students play a very important role in the experimentation in software engineering, as pointed out in [1], [10]. In situations in which the tasks to perform do not require industrial experience the experimentation with students is viable [9], [1]. The experiment owns also a point of strength: the number of subjects is approximately 100, and it seems to us discreetly valuable for statistical dependability.

As future work, we would like to include psychology research, at least in two directions: a more appropriate tools for knowledge capturing and controlling other psychological factors such as personality.

7. ACKNOWLEDGEMENTS

We’d like to thank Crescencio Bravo and Manuel Serrano for their help in executing the experiment. This research is supported by the MAS project partially supported by “DIRECCIÓN GENERAL DE INVESTIGACIÓN del Ministerio de Ciencia y Tecnología” (TIC 2003-02737-C02-02).

8. REFERENCES


9. APPENDIX

Figure 4. An exemplar Use Case

<table>
<thead>
<tr>
<th>1.1.1.1 Use Case</th>
<th>Send User Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1.1 Description</td>
<td>The Branch operator inserts data in the registration form, provided by the user. Validation of the form is launched.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>The form is not correct or complete. The sending of data is successfulness.</td>
</tr>
<tr>
<td>Actors</td>
<td>BranchOperator, HeadQuarterSystem.</td>
</tr>
<tr>
<td>Use Case Extends</td>
<td>Nn</td>
</tr>
<tr>
<td>Use Case Uses</td>
<td>Check Correctness/Completeness</td>
</tr>
<tr>
<td>Use Case Inputs</td>
<td>Name, address, offered books list (in case the user is a vendor) with specifications: title, author, publisher, language, publishing year, ISBN.</td>
</tr>
<tr>
<td>Use Case Outputs</td>
<td>Recording of data of the new user.</td>
</tr>
<tr>
<td>Criterion of Acceptance</td>
<td>Data of the new user are stored in the database of the Local Branch.</td>
</tr>
<tr>
<td>Related Expectations</td>
<td>Database management system. Correctness and completeness checks. Data sending to the Headquarter.</td>
</tr>
<tr>
<td>Related Reqs/Use Cases</td>
<td>Check Correctness/Completeness</td>
</tr>
</tbody>
</table>

Figure 5. Excerpt of Use Case specification

1. Could Remote Registration of User (User Remote Registration Sending) extend local user registration (User registration Sending)?
   a. Yes b. No and it does not make sense c. Possible, with proper modifications

2. Does the updating of user data (User Remote Registered Updating) require the correctness and completeness check (Correctness/Completeness Checker)?
   a. Yes b. No and it does not make sense c. Possible, with proper modifications

3. Could use cases Notification of Transaction To Buyer and Notification of Transaction To Branch be merged in one use case?
   a. Yes b. No and it does not make sense c. Possible, with proper modifications

4. Could the use case Update Database extends the use case Local Book Search?
   a. Yes b. No and it does not make sense c. Possible, with proper modifications

5. Given a transaction, can information concerning vendor be obtained through the Book (object)?
   a. Yes b. No and it does not make sense c. Possible, with proper modifications

6. A (object) Branch Operator must have executed at least one operation, otherwise it does not exist in the System.
   a. True b. False c. This information is not provided by documentation.

7. It is possible to obtain the list of registered users in a local branch through Data contained in a Branch (object).
   a. True b. False c. This information is not provided by documentation.

8. DataHandler (object) helps query the Database.
   a. True b. False c. This information is not provided by documentation.

9. Checker (object) verifies if all the fields of the form are filled in.
   a. True b. False c. This information is not provided by documentation.

10. The user interface is provided only for the remote part of the system.
    a. True b. False c. This information is not provided by documentation.

Figure 6. Questionnaire QB