Ranking and Rules for Selecting Two Persons in Pair Programming

Sultan Alshehri
Software Systems Engineering, Regina, Canada
Email: aljumais@uregina.ca

Luigi Benedicenti
Software Systems Engineering, Regina, Canada
Email: luigi.benedicenti@uregina.ca

***Abstract*— The analytic hierarchy process (AHP) has been applied in many fields and especially to complex engineering problems and applications. The AHP is capable of structuring decision problems and finding mathematically determined judgments built on knowledge and experience. This suggests that the AHP should prove useful in agile software development, where complex decisions occur routinely. This paper provides a ranking approach to help the XP team to set the rules of pairing two persons for pair programming and proposes several criteria can be used for the AHP evaluation. Two academic and the three-indusial case studies have applied the AHP to decide these rules in pairing.**

***Index Terms*— Extreme Programming; Pair programming; Analytic Hierarchy Process.**

I. Introduction

Pair programming in (XP) means that two programmers work together on one machine to do the same task. One of them is responsible for the typing the code (the driver); the second is responsible for watching and reviewing the problem currently being worked on (the navigator) [14]. XP programmers can achieve numerous of benefits when using pair programming, such as code with less defects, improved design quality, accelerated problem solving, timely delivery, fewer distractions and higher productivity [114].

This paper is organized as follows: section 2 present the current research on pair programming field; section 3 briefly explain the methodology of the work; section 4 explains the AHP method; the four pair programmers options are presented in section 5; four criteria for ranking the pair programmers alternatives are proposed in section 6; the case studies' results and findings are presented in section 7; section 8 discuss the validity of the study; and section 9 concludes the paper.

II. Current Research on Pair Programming

The current research is focusing on such areas as pair productivity, maximizing pair performance, evaluating the impact of pairs on the code quality and solving problems created when pair programmers have conflicts. Jan Hendrik et al. [3] provided assessment strategies to evaluate the individual programming abilities during pair programming situations. Tomayko [4] proved that when programmers work in pairs, they made fewer errors than in individual programming situations. VanDeGrift [5] found out that pair programming practice increases the programming performance and confidence. It also decreases the frustration levels of programmers. Also, pair programming could be a promising way of teaching the programming and elevate the programmers' skills.

Katira et al. [6] conducted a study involving 361 software engineering students at North Carolina State University to investigate the compatibility of pairs in pair programming. They found that students are compatible with partners whom they perceive of similar skill. They consider the midterm grades in class and the GPA to be skills indicators. The authors also found that mixing the genders pairs are less likely to be compatible. They stated "A collaborative style of programming seems to appeal more to female and minority students because of the highlighted social nature of the pair programming paradigm" [6].

Other work [7,9] examined the compatibility of the pairs among freshman, advanced undergraduate, and graduate students. They found that the students who have a partner in same skill level are more compatible than others. For example the graduate students work well with partners of similar actual skill level and a freshmen work better with partners with the same skill level.

Margolis [8] concluded that: "The feminine take on technology looks right through the machine to its social function, while the masculine view is more likely to be focused on the machine itself. As a result, when technology is introduced as an end in itself, as ill a programming class, for instance, young women are less likely to be interested than young men".

The National Centre for Education Statistics [10] shows a low representation of women and minority in computer science. Also, results of a survey-based study at the University of Wales [11] showed that pairs with lower self-esteem liked pair programming more than pairs with higher self-esteem. Also, Nelson [12] implemented "peer checking" experiment showed that the African-American

doi:10.4304/jsw.9.9.2467-2473

success rates improved when shifting from individual work to collaborative small groups. Salomon [13] states: "Knowledge is commonly socially constructed, through collaborative efforts toward shared objectives or by dialogues and challenges brought about by differences in persons' perspectives". Andrew and Bryan [14] emphasized that several personality traits should be considered when two developers are paired to collaborate effectively: effective communication, comfortableness working with a partner, confidence in one's abilities and the ability to compromise. Moreover, the initial findings indicate that pair programming produces shorter code (e.g. [14,15]) and results in better adherence to coding standards [16]. Müller [17] reported an increase of 5% on the total project costs caused by applying pair programming.

However, there does not appear to be a formal method to choose pairs in accordance with specific criteria. In this paper, we will show how AHP can be used to select the best pair matching based on the proposed criteria and among several alternatives.

III. Methodology

The study presented in this work is carefully designed to include two academic case studies and three industrial studies. This section describes in more detail the research question, unit of analysis and used data sources.

1) Research Questions

The primary objective of this study is to investigate how AHP can be used to decide the best pairs in pair programming. Moreover, the following research questions provided a focus for our case study investigation:

A. How can AHP help the XP team to match pairs based on specific criteria?
B. How do AHP results affect the developer's relation and performance?

2) Unit of Analysis

According to [18] the unit of the analysis should be defined from the main research questions of the study. The main focus is to rank several potential pairs to work together in coding. So the ranking and the process of evaluation are the two units of analysis for this study. Also, we consider the developer's view of how AHP benefits each XP practice. As result, this work is designed as multiple cases (embedded) with two units of analysis.

3) Data Collection and Sources

In the beginning of the study we propose the criteria affecting the ranking process and help to examine the AHP tool ability and benefits. This data was collected from literature review and previous studies. To increase the validity of this study, data triangulation was obtained. The data sources in this study were:

1. Archival records such as study plan from the graduate students.
2. Questionnaire given to the participants when developing the XP project.
3. Open-ended interviews with the participants.
4. Feedback from the customer.

The questionnaires and the open-ended questions only have been done with educational case studies.

4) Case Study Design

The educational case studies were performed as part of a course in the Advanced Software Design Class for graduate students taught in Fall 2012 at the University of Regina. The participants were 12 master's students and a client from a local company in Regina. Participants had various levels of programming experience and a good familiarity with XP and its practices. The Students' background related to the experiment includes several programming languages such as Java, C, C#, and ASP.net. They had implemented projects previously using various software process methodologies. The study was carried out throughout 15 weeks; students were divided into two teams. Both teams were assigned to build a project called "Issue Tracking System" brought by the client along with industrial requirements. It ran in 5 main iterations and by the end of the semester, the whole software requirements were delivered. The students were paired based on their experience and knowledge, but we also had an opportunity to pair some experts with novice and average programmers for the purpose of the study. Participants were given detailed lectures and supporting study materials on extreme programming practices that focused on pair programming rules. The students were not new to the concepts of XP, but they gained more knowledge and foundation specifically in the iteration plan, release planning and prioritizing the user stories. In addition, the students were exposed to the AHP methodology and learned the processes necessary to conduct the pairwise comparisons and to do the calculations. Several papers and different materials about the AHP and pair programming were given to the students to train them and increase their skills in implementing the methodology. In addition, a survey was distributed among students to get further information about their personal experiences and knowledge.

The researchers have visited three companies (two companies in Regina, and one in Calgary; both in Canada) several times and met with the developers and team leaders to explain the purpose of the study and to collect the data and feedback from the real industries. To preserve their anonymity the names have been withheld. All the companies are familiar with XP concept and currently practicing the pair programming during their development. In this study, eighteen experts have used their knowledge and average of 10 years experience to evaluate the proposed pair alternatives using the AHP.

IV. The Analytical Hierarchy Process Identify the Headings

AHP is a systematic approach for problems that involve the consideration of multiple criteria in a hierarchical model. AHP reflects human thinking by grouping the elements of a problem requiring complex and multi-aspect decisions [19]. The approach was developed by Thomas Saaty as a means of finding an effective and powerful methodology that can deal with complex decision-making

problems [20]. AHP comprises the following steps: 1) Structure the hierarchy model for the problem by breaking it down into a hierarchy of interrelated decision elements. 2) Define the criteria or factors and construct a pairwise comparison matrix for them; each criterion on the same level is compared with other criteria in respect of their importance to the main goal. 3) Construct a pairwise comparison matrix for alternatives with respect to each objective in separate matrices. 4) Check the consistency of the judgment errors by calculating the consistency ratio. 5) Calculate the weighted average rating for each decision alternative and choose the one with the highest score. More details on the method, including a step-by-step example calculation, are found in [19].

Saaty [8] developed a numerical scale for assigning the weight for criteria or alternative by giving a value between 1 (equal importance) and 9 (extreme importance), see table 1.

TABLE 1.
AHP NUMERICAL SCALE DEVELOPED BY SAATY.TABLE TYPE STYLES

Scale	Numerical Rating	Reciprocal
Equal importance	1	1
Moderate importance of one over other	3	1/3
Very strong or demonstrated importance	7	1/7
Extreme importance	9	1/9
Intermediate values	2,4,6,8	1/2, 1/4, 1/6, 1/8

V. PAIR PROGRAMMING OPTIONS

There are many ways of pairing programmers; Laurie Williams in his book "Pair Programming Illuminated" stated four possible alternatives that can be investigated: 1) Expert-Expert Pairing, 2) Expert-average Pairing, 3) Expert-Novice pairing, 4) and Novice- Novice Pairing. All of the possibilities of pairing have their own purposes and effects that can be summarized as follows:

1) Expert-Expert:

When pairing two experts there might be ego issues, but the work would benefit greatly. As Ron Jeffries states, "When the two experts get in sync, you can hear the lightning crackling. Working with a good expert partner is like gaining 40 or more IQ points" [21]. However, Lui and Chan conducted empirical experiment in pair programming and found that novice–novice pairs against novice solos are much more productive than expert–expert pairs against expert solos [22].

2) Expert-Average:

When expert pairs with average expert there is possible of raising his/her skill level. However, if the average person is not interested to expand his/her knowledge or doesn't interact well with the expert very well, that might create conflicts easily and defy the purpose of pairs.

3) Expert-Novice

The expert has to be willing to train the novice, which requires him/her to be more patient with slow development paces sometimes. On the other hand, the expert should welcome advice or suggestion from the novice and be able to admit the mistake if there is any.

4) Novice-Novice

This pairing is employed "[t]o produce production code in a relatively noncomplex area of the project, giving valuable experience to both programmers in the process" [21].

VI. PROPOSED CRITERIA FOR SELECTING THE OPTIMAL PAIRS

To find the best pair, it is necessary to determine the most important criteria that affect the participants when choosing the alternatives. The resulting criteria will be compared among each other based on the goal for importance. Finally, the potential pairs will be compared against each of the criteria [23]. In this paper, we propose four criteria that emerged during the case studies we conducted, but the method described in this paper can be applied to any set of criteria. The criteria shown below are simply illustrative of the decision method.

A. Speed: pairs with the highest chance to accelerate the coding practice;
B. Sharing Knowledge: pairs with the highest chance to exchange knowledge;
C. Code Quality: pairs with the highest chance to improve code quality more;
D. Learning: pairs with the highest chance to foster a training and learning environment.

1) AHP in Practice :

The first step in the analytic hierarchy process is to structure the problem as a hierarchy that includes three levels. The top level is the main objective: finding the best pairs; the second level is the criteria: speed, sharing knowledge, code quality and learning; the third level is the alternative: Expert-Expert Pairing, Expert-Average Pairing, Expert-Novice Pairing, Novice-Novice pairing. Fig.1 illustrates the AHP structure for the problem.

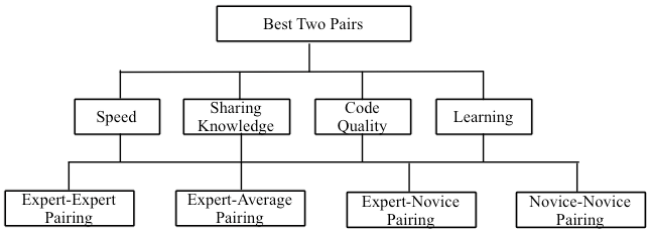

Figure 1. AHP structure for ranking the options of pairs.

Sheets of paper with appropriate AHP tables were handed to the all participants to keep the time short and facilitate the process of the comparison. The first page was dedicated to collecting general information about the evaluator, his/her experience, and the type and the level of his/her programming skills. A matrix is to compare the five criteria (C1: speed, C2: sharing knowledge, C3: Code quality, C4: learning).

VII. FINDINGS AND RESULTS

Each participant individually evaluated the pairs options based on the criteria mentioned earlier. The Expert Choice software [24] was used to calculate the aggregation results for the entire two teams collectively.

1) *Educational Case Studies Results*

For team 1, the ranking for the pairs based on all criteria, i.e. speed, sharing knowledge, code quality and learning, is summarized as follows. First: Expert-Expert (36.44); Second: Expert-Average (26.28); Third: Expert-Novice (21.74); Fourth: Novice-Novice (15.54). Table 2 summarizes the results.

The ranking for the best pairs by Team2 is summarized as follows: First: Expert-Expert (38.19); Second: Expert-Average (33.58); Third: Expert-Novice (19.59); Fourth: Novice-Novice (8.64). Table 3 summarizes the results.

TABLE 2.
PAIR PROGRAMMERS RANKING FOR TEAM 1

Pairs Ranking	Scores
Expert-Expert	36.44%
Expert-Average	26.28%
Expert-Novice	21.74%
Novice - Novice	15.54%

TABLE 3.
PAIR PROGRAMMERS RANKING FOR TEAM 2

Pairs Ranking	Scores
Expert-Expert	38.19%
Expert-Average	33.58%
Expert-Novice	19.59%
Novice - Novice	8.64%

Fig.2 shows the importance of each criterion as follows: code quality (56.11), sharing knowledge (23.94), learning (0.82), and speed (9.13).

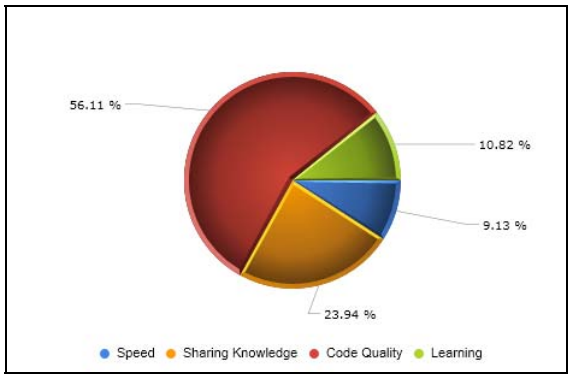

Figure 2. The Importance of the Crtiteria by Team1.

Fig.3 shows the importance of each criterion as follows: code quality (61.92), learning (15.64), speed (11.22), and sharing knowledge (11.22).

2) *The Industrial Cases Results*

To keep the companies anonymous, they will be called A, B, and C. The ranking for the pairs in the industrial environment is similar to the educational results with differences in the percentage ranking in each. The order for the alternative was as follows: First: Expert-Expert; Second: Expert-Average; Third: Expert-Novice; Fourth: Novice-Novice. Table 3 summarizes the results for each company.

TABLE 4.
PAIR PROGRAMMERS RANKING FOR COMPANY A

Pairs Ranking	Scores
Expert-Expert	39.29%
Expert-Average	24.96%
Expert-Novice	21.37%
Novice - Novice	14.38%

TABLE 5.
PAIR PROGRAMMERS RANKING FOR COMPANY B

Pairs Ranking	Scores
Expert-Expert	33.28%
Expert-Average	31.37%
Expert-Novice	27.96%
Novice - Novice	7.38%

TABLE 6.
PAIR PROGRAMMERS RANKING FOR COMPANY C

Pairs Ranking	Scores
Expert-Expert	35.97%
Expert-Average	28.18%
Expert-Novice	26.26%
Novice - Novice	9.59%

3) *Observations*

A. *Educational Cases:*

- Considering all the criteria together, both teams have the same ranking; the highest rank was expert-expert, the second expert-average, then expert-novice, finally novice-novice.
- Both teams considered code quality as the most important criteria. Sharing knowledge considered the second important criteria for team1 while team2 ranked the learning criteria the second important criteria.
- If we rank the best pairs considering each criterion individually, we can see both teams have ranked expert-expert the highest in terms of speed and code quality criteria, see tables 5 and 6.
- Team1 ranked the expert-novice the highest in terms of sharing knowledge and novice-novice in term of portability criteria, see tables 7 and 8.
- Team 2 ranked the expert-average the highest in terms of sharing knowledge and learning criteria, see tables 9 and 10.

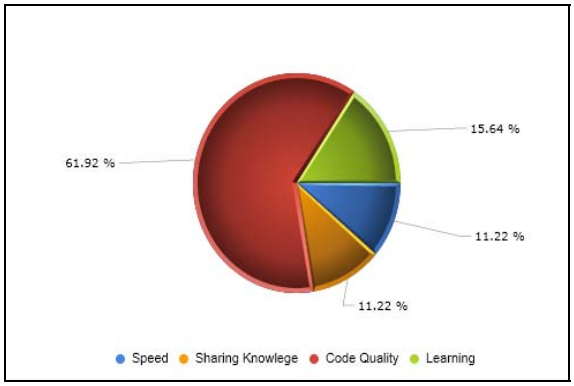

Figure 3. The Importance of the Crtiteria by Team2.

- When we did pairing expert-expert as the AHP results showed to be the highest, other issues raised up to the researchers, which is the conflict of opinions. In team 1, two experts had requested to change their pairs with others. However, team 2 has two expert as well without complaints. This indicates that there are other human factors that affect the interaction
- Also, when pairing experts with novices, the expert student felt he was doing most of the work. For the educational purposes, the students became concerned about the marks more than teaching others or sharing knowledge with others especially if they had limited time to submit the work.

B. Industrial Cases:

- Similar to the educational studies, the three companies have the same order of ranking.
- Code quality was considered the most important criterion for B and C companies, while A was considering the sharing knowledge criterion as the highest concern.
- If we look at the pair alternatives considering each criterion individually, we see that the three companies ranked the expert-expert in the top position in the speed and code quality criteria.
- For the sharing knowledge criteria, B and C ranked the expert-novice in the highest position, while A ranked the expert-expert in the top.
- In term of learning, A, B, and C ranked the expert-novice in the highest position.
- The expert in all the three companies admitted that the expert-expert pair is the best in reality as well, even though the conflict still exists everywhere. They confirmed that building relation through some social activities with the team members can strengthen the relation.

4) Semi-structured Interview Results

The semi-structured interview was conducted after showing the participants the results of the AHP evaluation for all the XP practices. Some of the results were surprising and others were expected. The interview included open questions to obtain students' general opinions about AHP, advantages and disadvantage of the using AHP, and the best experience for AHP among the all XP practices. As said previously, the data was collected in the form of handwritten notes during the interviews. These notes were organized in a folder to be analyzed and reached easily. The questions and answers for the semi-structured interview are below and the people names are kept anonymous:

From the interviews, we found that the AHP has received very positive feedback from the participants. AHP resolved the conflict of opinions of the process of pairing and brought every team members' voice to the decision in a practical way. It also empathized the courage among the team by letting every opinion be shared. The time and the number of the comparisons were the main concerns by the participants. All of them have recommended using the AHP in the future with the XP. Few recommendations such developing an automated tool to reduce the time for the AHP calculation, adding the mobility features, doing some cost and risk analysis, and trying it with other XP areas and studying the outcome.

5) Questionnaires

The questionnaires given to the participants were aimed to obtain the participants' perceptions and experiences with AHP. The questionnaires are divided into two main parts. The first part addresses questions about the AHP as a decision and ranking tool. The second part addresses questions regarding the direct benefit to the XP practice and investigating the participant's satisfaction. We used the seven-point Likert scale to reflect the level of acceptability impact about the AHP tool. The following are the meaning of the seven-point scale:

1. Totally unacceptable
2. Unacceptable.
3. Slightly unacceptable.
4. Neutral.
5. Slightly acceptable.
6. Acceptable.
7. Perfectly Acceptable.

After the participants answered the questionnaire, we calculated the results and presented the total percentage of the acceptability for each statement in the evaluation (questionnaires) in tables 4,5,6.

The total percentage of the acceptability was calculated as follow:

The total percentage of acceptability (TPA)

= The average of the score for each team * 100 / 7.

The average of the score for each team =

= The sum of the scores given by the team members / number of the team.

Table 7 shows the acceptability level for the AHP as a ranking tool.

TABLE 7.
ACCEPTABILITY LEVEL FOR THE AHP AS A RANKING TOOL

	Team1	Team2
AHP as a decision tool used in Extreme Programming (team 1, team 2)		
A- Decision Quality		
Capturing the needed Information	76%	88%
Clarity of the decision process.	88%	86%
Clarity of criteria involved.	81%	76%
Clarity of the alternatives involved	81%	79%
Goodness of the decision structure.	86%	90%
B- Practically		
Understandability	83%	88%
Simplicity	71%	86%
Time Efficiency	59%	62%
Reliability	74%	76%

The following questionnaires results for the impacts of the AHP to the process of selection

First: improving the team communication; team1 (83%) and team2 (90%).

- Second: creating a healthy discussion and learning opportunity, team1 (86%) and team2 (90%).
- Third: clarifying the ranking problem; team1 (83%) and team2 (93%).
- Fourth: resolving the conflict opinions among the members; team1 (78%) and team2 (93%).
- Fifth: increasing the team performance; team1 (79%) and team2 (94%).

VIII. VALIDITY

Construct validity, Internal Validity, External Validity and Reliability describe common threats of the validity of the performed study [25]. "Empirical studies in general and case studies in particular are prone to biases and validity threats that make it difficult to control the quality of the study to generalize its results" [26]. In this section, relevant validity threats are described. A number of possible threats for the validity can be identified for this work.

1) Construct Validity

This deals with the correct operational measures for the concept being studied and researched. The major threat to this study is the few number participated in each case study.

However, this threat has been mitigated using several techniques in order to ensure the validity of the findings.

• Data triangulation: a major strength of case studies is the possibility to use many different sources of evidence [25]. This issue has been taken into account through the use of surveys and interviews with different types of participations from different environments with various levels of skills and experiences, and through the use of several observations and feedback from the customer involved in the study. By establishing a chain of evidence, we could reach to our conclusion.

• Methodological triangulation: The research methods have been a combination of a real project conducted to serve this purpose, interviews, surveys, AHP results comparisons, and notes and researcher's observations.

• Member checking: presenting the results to the people involved in the study always recommended especially for the qualitative research. This is has been done by showing the final results to all the participants to ensure the accuracy of what was stated and to guard against researcher bias.

2) Internal Validity

This is only concerned about the explanatory case study [25] and it focused in establishing causal relationship. Students and educational constraints

This issue can be addressed by relating the research questions with other data sources providing information regarding the questions.

3) External Validity

This involves the domain of the study and the possibilities of generalizing the results. We address this by involving three companies to validate the ranking results. Even tough three companies had participated by putting their evaluation for pair programming, the sample size is very small: six experts from each company resulting in a total of 18 people involved.

Thus, there is the need to conduct more case studies in the industry involving more experts and developers to observe the similarities and the differences in findings.

4) Reliability

This deals with the data collection procedure and results. So, other researchers should arrive at the same case study findings and conclusions if they follow the same procedure. We address this by providing the research questions, case study set up, data collection and analysis procedure plan and execution steps and questionnaires.

IX. CONCLUSIONS

After using AHP to rank the pair programming alternatives, it was found to be an important tool that provides a very good vision for XP team when deciding how to create pairs. Considering the speed, sharing knowledge, code quality and learning when selecting the pairs could bring many advantages to the XP team, including the stakeholders. The relative weighting technique was the most preferable for both teams in our case studies, but the method we chose is general and thus the ranking can change depending on the team. More importantly, though, AHP helped students evaluate each pair option from different viewpoints. In addition, they could mathematically reconcile the conflict of opinions among them. The AHP introduces a cooperative decision making environment, which could accelerates the XP development process and maximizes the effectiveness of the software developed.

From the studies we conducted, we found also that even the results indicated that the expert-expert is the best pairs, other personalities and factors could play significant roles that may need efforts to compromise these factors and add them to the criteria to be ranked and evaluated.

REFERENCES

[1] J. Bevan, L. Werner, and C. McDowell, "Guidelines for the Use of Pair Programming in a Freshman Programming Class," in Proceedings of the 15th Conference on Software Engineering Education and Training, California University, Santa Cruz, CA, 2002.

[2] M. Matthias and F. Walter, "Extreme Programming in a University Environment," in Proceedings of the 23rd International Conference on, Page(s): 537-544,University Karlsruhe, Germany, May 2001. □

[3] Hahn, Jan Hendrik, Mentz, Elsa Meyer, and Lukas, "Assessment Strategies for Pair Programming," Journal of Information Technology Education, v8 p273-284, 2009.

[4] J. Tomayko, "A comparison of Pair Programming to Inspection for Software Defect Reduction," Computer Science Education, Vol. 12, No. 3, pp. 213-223, September 2002.

[5] T. VanDeGrift, "Coupling Pair Programming and Writing: Learning about Students' Perceptions and Processes,"

SIGCSE '04 in: Proceedings of the 35th SIGCSE technical symposium on Computer science education Pages 2-6, 2004.
[6] N. Katira, L. Williams, and J. Osborne, "Towards Increasing the Compatibility of Student Pair Programmers," in Proceedings of the 27th international conference on Software engineering Pages 625-626, 2005.
[7] N. Katira, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer, "On Understanding Compatibility of Student Pair Programmers," in Proceedings of the 35th SIGCSE technical symposium on Computer science education Pages 7-11, 2004.
[8] C. Rrunner, "Opening technology to girls," Scholaslic.com rcs. Rep, 1997, <http://scholastic.com/EL> (accessed 3.1.2013).
[9] J. Margolis, A. Fisher and F. Miller, "Caring about Connections: Gender and Computing," School of Computer Science, Carnegie Mellon University, IEEE Technology and Society Magazine, Winter, 1999/2000.
[10] National Centre for Education Statistics: Digest of Education Statistics, 1990-2002, Institute of Education Sciences, U.S. < http://nces.ed.gov/Programs/digest/> (accessed 4.10.2012).
[11] L. Thomas, M. Ratcliffe and A. Robertson, "Code Warriors and Code-a-Phoebes: a Study in Attitude and Pair Programming," SIGCSE Bull. 35, 1, 363-367, Jan2003.
[12] C. Nelson, "Student Diversity Requires Different Approaches to College Teaching Even in Math and Science," American Behavioral Scientist, vol. 40, pp. 165-175, 1996.
[13] G. Salomon, "Distributed Cognitions: Psychological and Educational Considerations (Learning in Doing: Social, Cognitive and Computational Perspectives)," Cambridge University Press; Reprint edition, December 28, 1996.
[14] A. Dick and B. Zarnett, "Paired Programming & Personality Traits," Red Hook Group, Toronto, Canada.
[15] W. Wood and W. Kleb, "Exploring XP for Scientific Research," IEEE Software, vol. 20, pp. 30 - 36, 2003.
[16] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," in the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000). 2001.
[17] M. Müller, "Are Reviews an Alternative to Pair Programming?," in the 7th International Conference on Empirical Assessment in Software Engineering, UK, 2003.
[18] R. Yin, "Case Study Research: Design and Methods," Second Edition, SAGE Publications, 1994.
[19] N. Tiwari, "Using the Analytic Hierarchy Process (AHP) to Identify Performance Scenarios for Enterprise Application", the Computer Measurement Group (2006).
[20] T. Saaty "The Analytic Hierarchy Process," McGraw-Hill, New York, (1980).
[21] L. Williams and Robert Kessler, "Pair Programming Illuminated," Addison-Wesley Professional; 1 edition, Jun 28 2002.
[22] K. Lui, K. Chan, "Pair Programming Productivity: Novice–Novice Vs. Expert–Expert," International Journal of Human-Computer Studies - Human-computer interaction research in the management information systems discipline archive Volume 64 Issue 9, Pages 915-925, September 2006.
[23] T. Saaty "How to Make a Decision: the Analytic Hierarchy Process," Interfaces, Vol. 24, No. 6, pp.19--43 (1994).
[24] Expertchoice for Collaborative Decision Making: http://www.expertchoice.com. (accessed 5.12.12).
[25] R. Yin, "Case Study Research – Design and Methods," 3rd edition, Sage Publications, London, 2003.
[26] R. Lincke, "How do PhD Students Plan and Follow-up their Work? – A Case Study," School of Mathematics and Systems Engineering, University Sweden.

Sultan Alshehri was born in Saudi Arabia in 1981; a PhD Student in Software Engineering Systems Department at University of Regina, Regina, Saskatchewan, Canada.

He worked as a computer science teacher in Riyadh, Saudi Arabia, 2004-2005. Currently, he is working as Programmer Analysts at SmarTech Company. In the same time, he holds the position of CEO for the LogicDots Company in Regina.

Mr. Alshehri holds a reward of a scholarship since 2005 until 2013 from the high Ministry of education in Saudi Arabia.

Dr. Luigi Benedicenti is the Associate Vice President (Academic) of the University of Regina, and a full professor in the Faculty of Engineering at the University of Regina. Dr. Benedicenti obtained his PhD in software engineering in 1999 from the University of Genoa, Italy.

He is a Professional Engineer licensed in Saskatchewan and a licensed Italian Engineer. His collaborative network extends beyond Saskatchewan with TRLabs and IEEE, and Canada through collaborative work with colleagues in Europe, South East Asia, and North America.

Dr. Benedicenti's current research is in three areas: Software Agents, Software Metrics, and New Media Technology. He envisions the unification of platform, tools, and optimizations for the provision of persistent distributed digital services, regardless of people's location and delivery device.