Institute of Mathematics and Statistics
University of São Paulo

Code Quality in Agile Methods: A Study on Group Development

Felipe Túlio Pereira da Cruz
Advisor: Alfredo Goldman
Co-advisor: Lucas Gren

São Paulo, November 2015

Summary

1 Abstract

Agile development process, popularized in the last decades, emerged as an alternative to traditional methods, based on the ”waterfall” life cycle, to ensure high software quality and faster development [1]. Since agile methods are mainly focused on group work, their performance may not always be the same [2]. Group development, a concept in the field of Psychology, states that a group pass through phases across time, dividing them in stages and relating some aspects to each one [3].

The agile methods assure better code quality [1], but is this related to the stage a group is? This study investigates the relationship between developers perceptions of effectiveness and productivity and the quality of the code that is produced by agile methods.

Data from students from the class of Laboratory of Extreme Programming at the University of São Paulo was gathered for this study. To assure Agile concepts, a total of 7 groups were made. Each individual answered a survey containing questions about their own perceptions regarding how efficient the group was and the quality of the code that were created. The data was analyzed and Pearson’s correlation was used to measure the relation between variables.

The findings show that code quality and group stage are not correlated. However, it was possible to notice bonds between quality and some aspects of productivity.

2 Introduction

At the beginning of the millennium, a group of preeminent software developers produced the Manifesto for Agile Software Development, which became known as Agile Manifesto [4], on which Agile Methods are based. Since then, those methods, as Scrum or XP, have been attracting increasingly developers for several reasons. Characteristics that draw attention for the approach are the capacity to cope with project changes through creation [1], delivering the final product early than a plan-driven procedure [5] or designing code with better quality [6]. Independent of the method adopted, the concept of team is crucial, because cooperation and self-organization are essential factors. However, groups go through stages, and groups perform differently in each stage [3]. Therefore, the quality of the application developed should be influenced by the stage the team is.

Group Development is a field in Psychology that studies how groups change over time. Since last century, a huge variety of models were proposed to explain those transformations. McGrath divides the stages into four modes, allowing groups to take their own path from the initial stage to the final one [7]. Tuckman and Jensen created their model starting from the review of about fifty studies, where a group trail a linear way (forming, storming, norming, and performing) [8]. A last example is Fisher's model, separating stages by how groups take their decisions [9]. A peculiar fact that most of the models has in common is that in certain stages a group has a superior performance than in other stages.

On the other hand, Agile Methods arose as an alternative to the well-established Waterfall Model. Mainly focused on the human factor, agile development has been growing on the market due to its capability to satisfy the software industry needs, delivering a superior, low-cost and faster product [10]. Jim Highsmith defines being Agile as able to ”Deliver quickly. Change quickly. Change often” [11]. Iterative production, focus on interaction, intensive communication between team members and the reduction

of artifacts are characteristics that Agile techniques share, despite the fact that each Agile approach has its own practices and priorities [10].

Having in mind that groups face the process of the changing stages across time to reach a desirable outcome and Agile teams rely on communication to develop their ideas, a bond between the efficiency related to the stage groups are and the quality of the software produced may exist.

It is possible to measure the productivity and performance of a group by the method proposed by Wheelan [12][13]. When it comes to software production, a few questions covering quality, maintenance and re-factoring were proposed. A correlation between both variables is desirable. If they are related, the code quality may vary with the stage the groups is. Otherwise, it is imaginable that a team can keep the quality of the software, independent of the stage it is. Also, correlations between individual questions can expose relevant aspects from the code developed and how efficient a group is.

In the next section, Related Work, we will review about the related literature, Group Development and Agile Methods. Right after, in Methods, the approach applied in this research, including data collection and analysis is described. The findings are shown in the Results section. Discussion about the outcome comes after. In the last section, everything is wrapped up and a Conclusion is made.

3 Related Work

3.1 *Agile Methods*

Because of the industry requirements and the advancement of the technology, traditional ways of developing software couldn't take over all the changes [10]. On a plan-driven methodology, extensive planning and systematized procedures are fundamental pillars; however, it may lead to predictable and/or repeatable actions [14]. To attend those demands, techniques focused on continuous and incremental cycles began to stand out [10].

In 2001, practitioners of those techniques gathered and proclaimed the Manifesto for Agile Software Development [4], a collection of lightweight methods in reply to the heavyweight plan-driven methods [15]. The manifesto states that Agile Development should rely on four core values:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

These four items are the foundation of the Agile Methodology, defining its values and distinguishing it from the traditional methods [10]. In contrast to traditional approach, Agile methods are characterized by: collaboration, inside the development group and outside of it; code reviews, granting the diffusion of key information; small teams, requiring less effort to coordinate the team; short iterations, turning changes in the project easier; timeboxing, improving productivity; and continuous testing, ensuring performance [1].

3.2 *Extreme Programming*

One of the first emergent Agile Methods, Extreme Programming was introduced by Kent Beck in 1999 and quickly adopted by developers. Beck describes XP (Extreme Programming) as ”a style of software development focusing on excellent application of programming techniques, clear communication and teamwork” [16].

XP is identified as having 12 practices during the development process. Practitioners recognize that each practice by itself does not strengthen the methodology, but their combination does [10]. The twelve practices are defined as follows [16]:

- *Planning Game:* in the beginning of each iteration, developers and customers estimate and determine requirements for the next release;
- *Small Releases:* small versions of the systems released often that assist in the acceptance process by the customer;
- *Metaphor:* customers and developers create stories that model the system;
- *Simple Design:* keep the design as simple as possible;
- *Tests:* developers constantly write acceptance, unit and functional tests, which must run impeccably;
- *Refactoring:* the process of restructuring the system maintaining its behavior, reducing code complexity and improving readability;
- *Pair Programming:* two developers at the same machine write all code;
- *Continuous Integration:* new features are integrated into the system once it is developed and tested;
- *Collective ownership:* all developers own the code, and anyone can make changes if necessary;

- *On-site Customer:* a customer is part of the team to answer questions;
- *40-hour Weeks:* work with quality in a healthy pace;
- *Coding Standards:* all the code produced follows stipulated patterns.

3.3 *Code Quality*

Software quality is difficult to define, represent, comprehend and measure, even though it is crucial for a product success [17]. According to the Institute of Electrical and Electronics Engineers (IEEE), quality is: ”The degree to which a product or process meets established requirements; however, quality depends upon the degree to which those established requirements accurately represent stakeholder needs, wants and expectations” [18].

As an attempt to maintain a satisfying quality, several methods are used to improve code condition. Refactoring is a technique where small changes are made to the code, turning its design better without changing its external behavior. The reconstruction of the code is based on code checkup, reducing error probability [19]. Another method is Software Maintenance, which is the correction of defects and fixing faults to improve performance.

3.4 *Group Development*

A field of Psychology, entitled Group Development, investigates the behavior of small groups and how they change over time, the manner they are formed, their work style and the moment of dismantle [20]. Aspects analyzed include the quality of the output produced, its cohesiveness, existence of conflict and the frequency of activities. Patterns of change and if there is continuity in their conduct are examined to generate models.

According to Bales, a definition of group is: ”A small group is defined as any number of persons engaged in interaction with each other in a single

face-to-face meeting or series of meetings, in which each member receives some impressions or perception of each other member distinct enough so that he can, either at the time or in later questioning, give some reaction to each of the others as individual person." [21].

Through time, in order to remain effective and keep their development groups must perform crucial task: they should achieve their objectives, keep a good environment between group members and be capable to readjust to unexpected situations [22].

3.5 *Integrated Model of Group Development*

The integrative model of small group development, presented by Wheelan, takes the perspective that a group achieves maturity as it members keep working together [12]. This model associates distinct conducts and standards of a group to a specific stage. A group's life, although linear in a sense, is divided into 4 stages, where it evolves over time passing through them.

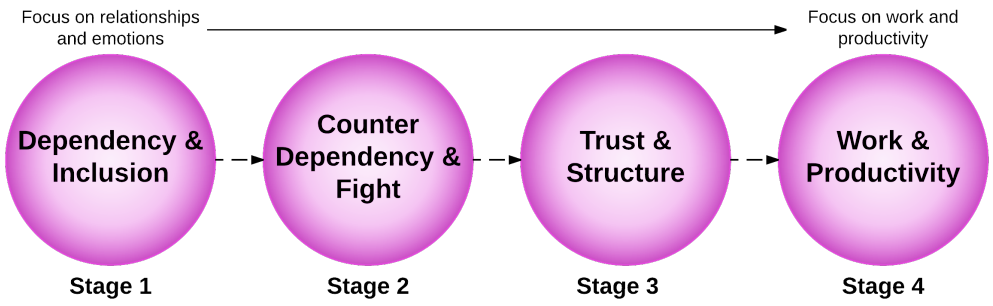

Figure 1: Group Development Stages

Stage 1 - Dependency and Inclusion: The first stage of this model is represented by significant dependency on the nominated leader, preoccupation about security and inclusion problems. In this stage, members rely on the leader to give instructions and determine in which direction the group should go [3].

Stage 2 - Counter Dependency and Fight: In this stage, conflict is inevitable. Members disagree with themselves as they present different point of views or express contradictory opinions. Nevertheless, to build the group's confidence and create an environment where members are comfortable to discord, disagreement is fundamental [3].

Stage 3 - Trust and Structure: Conflicts continue to occur in this stage; however, the groups is mature enough to manage its conflicts more effectively. A boost of cohesion, confidence and tolerance arise due the positive working relationships formed. Subgroup is encouraged and division of labor takes place [3].

Stage 4 - Work and Productivity: The fourth and the last stage is characterized by intense team work and efficiency. The group gets, gives and uses feedbacks about its productivity and effectiveness. The group finally is able to focus on goal achievement, since many issues of the preceding stages where resolved [3].

3.6 *Group Development Questionnaire (GDQ)*

In order to measure a group's stage in the Integrated Model of Group Development, Wheelan designed the Group Development Questionnaire [13]. The survey was divided into 4 sections, each section was arranged to evaluate the amount of energy expended on a stage individually.

Test questions of each section were composed to recognize the existence or lack of behaviors from the stages of development. The items on the first section measure how much energy is spent dealing with problems about reliance and involvement. The second section is related to how intense is the group concentration on conflicts and counter dependency problems. Questions from the third section seek to evaluate the degree of trust and structure presented by the group. Aspects from stage 4 are estimated by the last section [3]. Sample items can be found below:

Section	Sample Items
GDQ 1	Members tend to go along with whatever the leader suggests
	We haven't discussed our goals very much
GDQ 2	Members challenge the leader's idea
	There is quite a bit of tension in this group at this time
GDQ 3	We can rely on each other. We work as a team.
	The group spends its time planning how it will get its work done
GDQ 4	The group acts on its decisions
	This group encourages high performance and quality work

Table 1: GDQ sample questions from each section

There are a total of 60 questions, 15 for each stage. Each item is rated from 1 (never true for this group) to 5 (always true for this group). Therefore, the minimum score for a section is 15, and the maximum is 75.

The fourth section evaluates work and productivity (GDQ 4) and has been proven to correlate with a variety of effectiveness measures in different sectors. For example, intensive care faculty saves more lives [23] and high school staff leads to a better performance of their students [24] when their GDQ 4 score is higher.

4 Method

4.1 *Participants*

The sample for this study was obtained in the class of Laboratory of Extreme Programming at the Institute of Mathematics and Statistics, in the University of São Paulo. The goal of Lab XP is to introduce the concept of Agile Methods through Extreme Programming, by developing a software project for a stakeholder.

Students voted for the most interesting proposals from a body of clients. Afterwards, groups were formed depending on the personal choice of each student, that were supposed to choose 3 proposals to work in. The class contained 40 students that were divided into 7 groups. Group size varied from 6 to 8 members, including a preselected coach, a student that already had contact with Agile Methods before.

Each group were able to select in which programming language they would develop and how frequent meetings would be; as long as they comply with the norms established by the Extreme Programming method and gather together for a minimum of 8 hours per week to develop as a group.

4.2 *Survey*

Every student received a survey to answer that were composed by the 15 questions of GDQ 4, to measure how productive and efficient groups were, and 3 question concerning the quality of the code produced. The 18 questions from the survey were rated on a Likert scale from 1 (low agreement to the statement) to 5 (high agreement to the statement). Next are the three questions related to code quality.

How would you rate the code quality in your product(s)?
How often does the code need maintenance?
How much ”extra time” is given for cleaning up and re-factoring the source code?

Table 2: Code Quality Questions

4.3 *Procedure*

The surveys were distributed to every participant during class time, after a initial period of about a month. Groups varied in the amount of code produced up that moment, some had a substantial quantity and some had just restarted their work. They had a couple of days to fill it out. After the deadline, the surveys were collected from the coach of each team. Every survey was answered; however, a few were not complete.

4.4 *Data Analysis*

In order to verify if there is a link between how groups perceive their productivity and efficiency with the quality of the code developed, Pearson's Correlation analysis was performed. Pearson's Correlation measures the strength of the association between two variables. Likewise, Pearson's Correlation was adopted to check the existence of a correlation between each question from the GDQ 4 section with each question from the code quality section.

However, a number of researchers have described complications associated with studying groups from a Social Science perspective. For that reason, the Significance generated by Pearson's Correlation will be interpreted differently [25]. Cohen analyzed the persistent neglect of statistical analysis in behavior studies and designed new threshold values for standard statistical tests based on his ideas of effect-size [25].

Pearson's Correlation , denoted by *r*, vary from -1 to 1, where positive values denote positive linear correlation and negative values denote negative linear correlation. A value of 0 denotes no linear correlation. A table with the standard interpretation of correlation and Cohen's idea of effect-size can be observed as follows.

	Weak	**Moderate**	**Strong**
Traditional Interpretation	0.4	0.7	1
Cohen's Idea of Effect Size [25]	0.1	0.3	0.5

The level of significance for two-tailed tests was set to 0.05. Still, the size of the sample needs to be taken into consideration. Because of the small sample, moderate correlations may misleadingly not reach significance. Therefore, the value of correlations must be interpreted carefully.

Since questions 2 and 3 from the code quality section are negatively worded, it is not possible to use the same scoring as positively worded questions. Therefore, the reverse of their scores were used to produce the results. In other words, to generate the correlations those questions had their results reversed, i.e., rate 1 was designated 5; rate 2 was designated 4; and so forth.

A Quantile-Quantile plot of Standardized Residuals (Q-Q plot) is a graphical technique for determining if two data sets come from populations with a common distribution by plotting their quantiles against each other. A Q-Q plot was calculated with the purpose of evaluating whether the data was normally distributed or not. It is possible to verify, in Figure 2, that the points are close to the reference line, which indicates that the sample seems to come from a normal population.

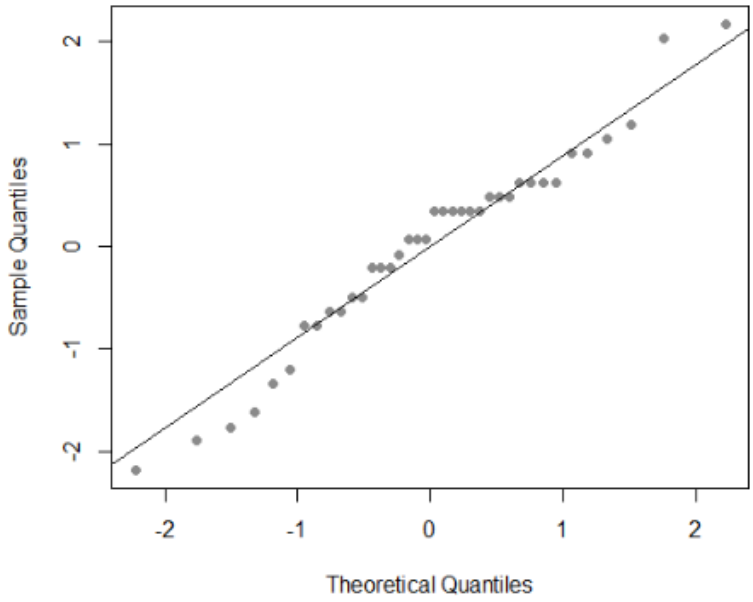

Figure 2: Q-Q plot of Standardized Residuals calculated from the regression model

The Shapiro-Wilk test checks whether a random sample comes from a normally distributed population or not by the usage of the null hypothesis principle. The null hypothesis asserts that the population is normally distributed. Thus if the null hypothesis is rejected, there is evidence that the data tested are not from a normally distributed population. Otherwise, the null hypothesis that the data came from a normally distributed population cannot be rejected. From this test, it is possible to conclude that there is no evidence to testify, with a 5% confidence level, that the sample from this study was not retrieved from a normal population.

5 Results

5.1 *Correlation of individual questions*

The outcome obtained by Pearson's Correlation between individual questions from GDQ 4 and each question from code quality (CQ) section and the respective significance can be verified below:

Person's Correlation	**CQ 1**	**CQ 2 rev.**	**CQ 3 rev.**
Significance (2-tailed)	**(N = 39)**	**(N = 37)**	**(N = 36)**
GDQ4 1	0.059	0.094	-0.109
(N = 40)	0.719	0.578	0.523
GDQ4 2	0.291	0.347	0.045
(N = 40)	0.072	0.035	0.794
GDQ4 3	0.498	0.097	-0.076
(N = 40)	0.001	0.568	0.657
GDQ4 4	0.160	0.226	-0.254
(N = 40)	0.331	0.177	0.134
GDQ4 5	0.299	0.311	0.041
(N = 40)	0.064	0.061	0.0812
GDQ4 6	0.153	0.277	0.023
(N = 40)	0.353	0.097	0.892
GDQ4 7	0.331	0.234	-0.162
(N = 40)	0.039	0.163	0.345
GDQ4 8	0.488	0.318	-0.087
(N = 40)	0.001	0.055	0.615

Person's Correlation	**CQ 1**	**CQ 2 rev.**	**CQ 3 rev.**
Significance (2-tailed)	**(N = 39)**	**(N = 37)**	**(N = 36)**
GDQ4 9	0.226	0.257	-0.056
(N = 39)	0.172	0.130	0.751
GDQ4 10	0.264	0.225	0.167
(N = 40)	0.104	0.180	0.331
GDQ4 11	0.309	0.253	-0.290
(N = 40)	0.056	0.130	0.86
GDQ4 12	0.348	0.141	-0.319
(N = 39)	0.032	0.412	0.062
GDQ4 13	0.136	0.143	-0.327
(N = 40)	0.410	0.398	0.052
GDQ4 14	0.136	0.170	-0.224
(N = 39)	0.414	0.322	0.195
GDQ4 15	0.022	-0.119	-0.497
(N = 40)	0.896	0.482	0.002

Table 3: Correlation between individual questions

The results can be divided into three sections depending on the significance (2-tailed) from each correlation. Significance values above from 0.06 until 0.1, results that can't be characterized as not statistically significant, because of the sample size. A range above 0.05 to 0.06 for the significance values that have a rate close to the necessary acceptance. Finally, the significance values from 0 to 0.05, the ones that have strong evidence to reject null hypothesis.

From the first section, it is relevant to point out the following results:

- *CQ 1 / GDQ4 2:* Approximately moderate correlation stating that if the group is accomplishing some of its goals, superior code quality is expected;
- *CQ 1 / GDQ4 5:* Roughly moderate correlation asserting that the better is the group decision making approach, the better is the code quality;
- *CQ 2 rev. / GDQ4 5:* Moderate correlation affirming that the better is the group decision making approach, the less code maintenance is needed;
- *CQ 2 rev. / GDQ4 6:* Weak correlation affirming that the more participatory decision making methods are, the less maintenance is demanded;
- *CQ 3 rev. / GDQ4 11:* Relatively moderate correlation stating that the more attention given to details, the more refactoring is performed;
- *CQ 3 rev. / GDQ4 12:* Moderate correlation declaring that if the group maintains good relationship with other groups, more refactoring is observed.

Outcomes from the second section are listed below:

- *CQ 1 / GDQ4 11:* Moderate correlation stating that additional attention devoted to details, the better is the code quality;
- *CQ 2 rev. / GDQ4 8:* Moderate correlation affirming that if high performance and quality work is encouraged, the less code maintenance is required;

- *CQ 3 rev. / GDQ4 13:* Moderate correlation asserting that if technical and people resource are utilized, code refactoring is more present.

Finally, results from the last section are described as follows:

- *CQ 1 / GDQ4 3:* Practically strong correlation affirming that greater use of feedbacks about the group effectiveness and productivity, high quality code is predicted;
- *CQ 1 / GDQ4 7:* Moderate correlation stating that if the group acts on its decisions, the code quality is better;
- *CQ 1 / GDQ4 8:* Relatively strong correlation asserting that the more high performance and quality is encouraged by the group, code tends to be of high quality as well;
- *CQ 1 / GDQ4 12:* Moderate correlation declaring that if the group has good relationships with whom it interacts, code quality is expected to be high;
- *CQ 2 rev. / GDQ4 2:* Moderate correlation asserting that if goals are being accomplished, the less code maintenance is demanded;
- *CQ 3 rev. / GDQ4 15:* Basically strong correlation stating that if subgroups is encouraged, code refactoring is performed more often.

5.2 *Correlation between code quality and GDQ 4 sections*

The observed value for the Pearson's Correlation between both variables is 0.277, which represents a moderate positive linear correlation between them. The coefficient of determination (r^2), acquired from the regression analysis, had a result of 0.0767. This value indicates that, approximately, 7.67% of variance in the code quality is explained by the GDQ4 measurements.

The Analysis of Variance (ANOVA) Test, a statistical method used to test differences between two or more means, was ultimately performed. This test indicates with a *p*-value of 0.0922, that the perception of productivity and efficiency and the quality of the code developed does not have a significant difference.

6 Discussion

6.1 *Correlation of individual questions*

Outcomes from this study revealed significant relationships between distinct variables. Furthermore, empirical evidence that was taken for granted in the field of Computer Science, may start being supported.

With regard to developing code with high quality, it is strongly positive correlated to some practices that groups perform. The code excellence was proven to be better if the development group reinforce high performance and quality. Moreover, the usage of feedbacks to notify the group itself about how effective and productive they are also reflects positively on the code developed. Other moderate correlations are taken in consideration in this parameter. The accomplishment of goals, the decision making proposal and dedication to details leads to a better code.

Considering the amount of maintenance demanded, moderate correlations, but positive oriented, were discovered. The main observation is the correlation that indicates the less need for maintenance thanks to goal attainment. However, other results can't be neglected, as the correlation with participatory decision-making approaches and the encouragement of high performance and quality.

Concerning code refactoring, a negatively strong correlation stands out among the rest. If the division into subgroups is encouraged, refactoring is more frequent. Nevertheless, the usage of technical and people support, good relationship with other individuals or groups and excessive care spent in details contributes to extra time refactoring code.

It is difficult to relate findings from this experiment to previous researches, since the field of Agile Development is relatively new and the relation with Psychology is being established gradually.

6.2 *Correlation between code quality and GDQ 4 sections*

This outcome comes against the odds, since other researches based on the measurement of work and productivity correlating it with a diversity of effectiveness measures in various sectors indicated positive correlations[23][24].

Nonetheless, this finding still is valuable. The information that code quality developed by a group adopting Extreme Programming, an Agile Method, is not related to how much the group perceive their efficiency and productivity, the GDQ 4 measurement, brings another perspective to the software development scenario. It implies that quality of the code produced doesn't depend on the group's efficiency and productivity, i.e, a group that doesn't consider itself efficient and productive may develop a high quality code, and vice-versa.

The impact of this result suggest not to focus on the development of the group to produce better code. Instead of spending time and energy in a variable that will not influence in a superior outcome, other aspects should deserve more attention, as the ones mentioned previously.

7 Conclusion

This research shows a correlation between the quality of the code developed by groups applying an Extreme Programming approach and the awareness of how efficient and productive the groups are. In addition, a further examination of specific aspects from both concepts, Agile Methods and Group Development, was performed to verify the existence of correlations. Through a correlation analysis, a direct link between the effectiveness and productivity of a group and the quality of what was developed was not encountered. On the other hand, connections from particular characteristics of those variables were found, such as: code quality depends on the usage of feedbacks and on the encouragement of high performance and quality work; accomplishment of goals leads to less code maintenance; and if subgrouping is stimulated, code refactoring is executed more often. These findings are important to agile software development, since information from different perspectives, such as Psychology, may help discovering means to improve the outcome of it.

The present study is a step towards a better understanding of assumptions that are present in the development environment. In terms of future research, a larger sample of agile developers already introduced to the software development scenario is favored. Also, a more precise method to measure the quality of the code developed, instead of a self opinion about the matter. Thus, an additional research with the conditions mentioned earlier to support or refute this study is recommended.

8 Acknowledgements

Special thanks to Alfredo and Lucas, who instructed me from the beginning of this work and gave me assistance and advice to complete it. Also to the friends I made through this journey, without them I'd be lost in the middle of it.

Thanks to my family and my chayote that I was able to accomplish one of the most challenging and arduous goals I set to my life. Only them know how much I donated myself to achieve it. I'd like to thank my parents, because they gave everything I needed to be where I am. Also my siblings, those that I'll support until the end. Molly, my beloved dog. And specially Bianca, the person that gave me love, care and made my dreams come true.

9 References

[1] M. Coram and S. Bohner, *The Impact of Agile Methods on Software Project Management*, in International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05). IEEE, 2005.

[2] C. Tolfo, R. Wazlawick, M. Ferreira and F. Forcellini, *Agile Methods and Organizational Culture: Reflections about Cultural Levels*, in Journal of Software Maintenance and Evolution: Research and Practice. Software Process: Improvement and Practice, pp. 423-441, 2009.

[3] S. Wheelan, B. Davidson and F. Tilin, *Group Development Across Time: Reality or Illusion?*, in Small Group Research. Vol. 34, No. 2, pp. 223-245, 2003.

[4] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherloand and D. Thomas, *Manifesto for Agile Software Development*, in www.agilemanifesto.org. Retrieved in October 2015.

[5] J. Manzo, *Odyssey and Other Code Science Success Stories*, in CrossTalk. Vol. 15, No. 10, pp. 19-21 & 30, 2002.

[6] T. Dybå and T. Dingsøyr, *Empirical Studies of Agile Software Development: A Systematic Review*, in Information and Software Technology. pp. 44-53, 2008.

[7] J. McGrath, *Time, Interaction and Performance (TIP) A Theory of Groups*, in Small Group Research. Vol. 22, No. 2, pp. 147-174, 1991.

[8] B. Tuckman and M. Jensen, *Stages of Small-Group Development Revisited*, in Group & Organization Studies. pp. 419-427, 1977.

[9] B. Fisher, *Decision emergence: Phases in group decision making*, in Communication Monographs. pp. 37 & 53-66, 1970.

[10] D. Cohen, M. Lindvall and P. Costa, *An Introduction to Agile Methods*, in Advances in Computers. Vol. 62, 2004.

[11] J. Highsmith, K. Orr and A. Cockburn, *Extreme Programming*, in E-Business Application Delivery. 2000.

[12] S. Wheelan, *Group Processes: A developmental perspective*, in Boston: Allyn & Bacon. 1994.

[13] S. Wheelan, *The Group Development Questionnaire: A manual for professionals*, in Provincetown, MA: GDQ Associates. 1994.

[14] B. Boehm, *Get Ready for Agile Methods, With Care*, in IEEE Computer. IEEE, 2002.

[15] C. Larman and V. Basili, *Iterative and Incremental Development: A Brief History*, in IEEE Computer. IEEE, 2003.

[16] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, Boston, 2003.

[17] B. Kitchenham and J. Walker, *A Quantitative Approach to Monitoring Software Development*, in Software Engineering Journal. pp. 2-14, 1989.

[18] IEEE, *IEEE Standard for Software Quality Assurance and Processes*, in IEEE Std 730-2014. 2014.

[19] P. Sfetsos and I. Stamelos, *Empirical Studies on Quality in Agile Practices: A Systematic Literature Review*, in 2010 Seventh International Conference on the Quality of Information and Communications Technology. IEEE, 2010.

[20] A. Adnan, A. Akram and F. Akram, *Group Development: Theory and Practice*, in Middle-East Journal of Scientific Research. Vol 16, No. 10, pp. 1428-1435, 2013.

[21] R. Bales, *Interaction Process Analysis: A Method for the Study of Small Groups*, in Addison–Wesley Press, Inc. 1950.

[22] D. Johson and F. Johson, *Joining Together: Group Theory and Group Skills*, in A and B Publishing. 8th ed. 2003.

[23] S. Wheelan, C. Burchill and F. Tilin, *The Link Between Teamworks and Patients' Outcomes in Intensive Care Units*, in American Journal of Critical Care. Vol. 12, No. 6, pp 527-534, 2003.

[24] S. Wheelan and F. Tilin, *The Relationship Between Faculty Group Development and School Productivity*, in Small Group Research. Vol. 30, No. 1, pp 59-81, 1999.

[25] J. Cohen, *Quantitative Methods in Psychology*, in Psychologycal Bulletin. Vol. 112, No. 1, pp 155-159, 1992.