A Case Study On Teaching Agile Software Development Methods

A. Spiteri Staines

Abstract— This paper presents and discusses some interesting findings from teaching Agile methods and modelling in a typical university at an undergraduate level software engineering course. The students who took part in this study had been exposed to a few hour lectures on Agile methods and other traditional methods like the RUP (rational unified process). A simple modelling questionnaire was presented to the students. The results were recorded, analysed and interpreted. Findings are discussed and conclusions are given.

Index Terms— Agile Methods, Agile Modelling, Extreme Programming (XP), Best Practice Approaches, Requirements Elicitation, Requirements Engineering, Software Engineering, Software Modelling, Software Quality Assurance, Teaching Software Engineering

1 Introduction

THE past decades have seen the emergence of many new methods used for software engineering. Teaching these methods to students at undergraduate level is by no means a simple feat. Years of hands on experience in the field cannot be obtained from reading books. This is even more evident when light-weight approaches like Agile methods are taken into account [1]-[3]. Agile methods have to be taught after the students have a sound foundation in more traditional methods and notations. Agile methods can be mainly presented to students at a surface level because indepth knowledge would require several years of experience in the field. This cannot be obtained from just a university course. These methods are based on certain key principles and ideologies that have to be part of the organizational setting, where they are being employed. Teaching and instructing students on key principles of software engineering is very important for their future development. Teaching these methods is not so simple as the method itself. This is because acquaintance with traditional and rigourous methods is a must before comprehension about these lightweight methods is possible.

Software engineering is a vast subject that is dependent on other areas like requirements engineering, requirements elicitation and other topics like formal methods. Those teaching key principles in software engineering must strike a balance between formal and informal techniques [1]-[7].

Rapid application development and Agile methods have become commonplace in the modern world. Agile development methods are also known as lightweight methods. These development styles are focused to respond to changing environments and getting software application development done on time for particular business organizations. Agile methods are particularly well suited to small to medium organizations that focus on business and E-Commerce. Agile methods are prescribed for business organizations where requirements are frequently changing over a period of time. Evaluation of software takes place rapidly because of time constraints. Systems development takes place as a series of versions that are discussed with regular meetings with stakeholders [5]-[7].

Agile methods have been developed to deal with the issues and problems related to more traditional approaches. The very word Agile implies focusing on doing things or getting things done rather than just focusing on design only. Agile development is a continuous process of self improvement based on iterations. The idea is to deliver high quality quick solutions at a low cost. However, the success completely depends on the organization. Agile methods come from a long lasting cultural setup in the organization.

Teaching Agile methods to students is cumbersome: i) Hands on exposure to notations and methods is missing. ii) Some form of formal background and understanding the importance of software development methods does not exist unless there is previous workplace experience. iii) Important agile principles cannot be transferred from a course. These can only be understood in a work environment. Some important Agile development principles like: i) priority to satisfy the customers, ii) team collaboration, iii) good communication principles that must exist in the organization, etc. are imperative for the success of Agile.

Some consider that the Agile principles are anti-methodology and require no diagrams or notations at all. This is a mistaken idea, because Agile tries to create a balance in this respect. Modelling is important, but models must not be created just for the sake of creating them. Their use is for helping the stakeholders comprehend the scenario.

This study deals with examining how students comprehend and view Agile modelling in a small teaching environment at undergraduate level.

2 Methodology

2.1 How the Study was Conducted

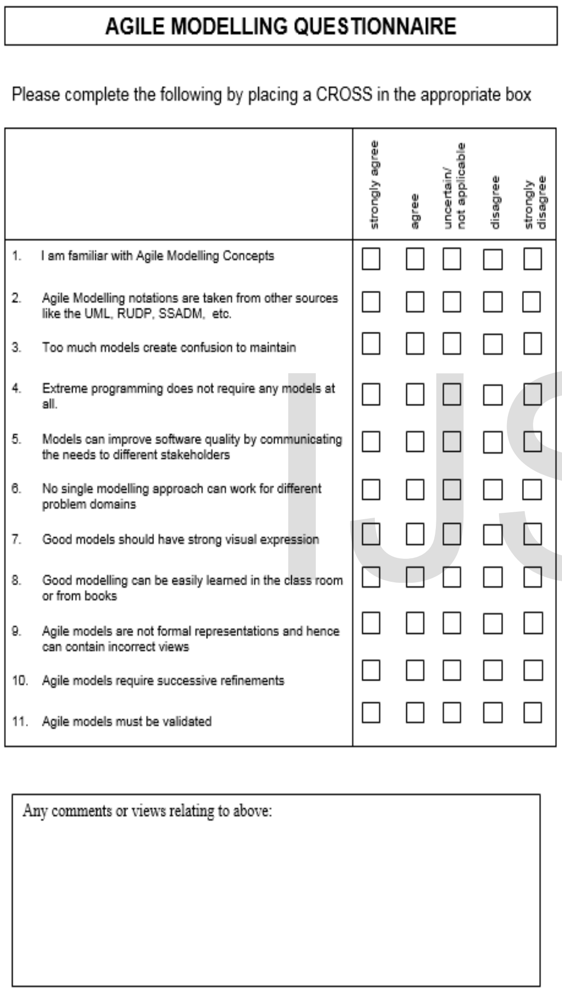
A small group of students in an undergraduate software engineering course were selected for this study. The group consisted of about 30 students. These were presented with a structured questionnaire after they had been exposed to Agile methods like Scrum and XP (extreme programming). The students were left free to answer or ignore the questionnaire.

• *Anthony (Tony) Spiteri Staines is an Associate Academic at the Department of Computer Information Systems at the Faculty of ICT, University of Malta, e-mail: toni_staines@yahoo.com*

Out of the 30 students 13 students responded to the questionnaire in full and handed in their results.

2.2 Questionnaire Used

The questionnaire used for this study consisted of eleven questions. The questions were based on the following: i) Familiarity with Agile Modelling Concepts. ii) The source from where the notations are derived. Thus it is possible to have notations from other sources like the UML, RUDP, SSADM, etc. iii) The confusing problems of having too many models, iv) The assumption that XP (extreme programming) does not require any models at all, v) The assumption that models can improve software quality by communicating the needs to different stakeholders, vi) The idea that no single modelling approach can work for different problem domains, vii) Good models should have strong visual expression, viii) Modelling can be easily learned in the class room or from books, ix) The models are not formal representations and hence can contain incorrect views, x) The issue that models require successive refinements, xi) The requirement for validation.



AGILE MODELLING QUESTIONNAIRE

Please complete the following by placing a CROSS in the appropriate box

	strongly agree	agree	uncertain/ not applicable	disagree	strongly disagree
1. I am familiar with Agile Modelling Concepts	☐	☐	☐	☐	☐
2. Agile Modelling notations are taken from other sources like the UML, RUDP, SSADM, etc.	☐	☐	☐	☐	☐
3. Too much models create confusion to maintain	☐	☐	☐	☐	☐
4. Extreme programming does not require any models at all.	☐	☐	☐	☐	☐
5. Models can improve software quality by communicating the needs to different stakeholders	☐	☐	☐	☐	☐
6. No single modelling approach can work for different problem domains	☐	☐	☐	☐	☐
7. Good models should have strong visual expression	☐	☐	☐	☐	☐
8. Good modelling can be easily learned in the class room or from books	☐	☐	☐	☐	☐
9. Agile models are not formal representations and hence can contain incorrect views	☐	☐	☐	☐	☐
10. Agile models require successive refinements	☐	☐	☐	☐	☐
11. Agile models must be validated	☐	☐	☐	☐	☐

Any comments or views relating to above:

Fig. 1. A Sample of the Agile modelling questionnaire handed out to the students. The scoring was simply done by ticking the boxes to indicate the approximate value that the student identifies with.

The results in the questionnaire are ranked accordingly to the Likert scale: i) Strongly agree, ii) Agree , iii) Uncertain/not applicabe , iv) disagree and v) strongly disagree with the proposed statements,by placing a cross or ticking the appropriate box. The basic structure of this questionnaire is depicted in fig. 1.

2.3 How the Results Were Obtained

When the students returned the questionnaire it was analysed using very basic measures to understand the actual data obtained. The method used to analyze the data was quite primitive. The data was loaded into a spread sheet, where a simple evaluative ranking scale was used. Stongly Agree 5, Agree 4, Uncertain/Not Applicable 3, Disagree 2 and Strongly Disagree 1.

The scores for those who answered the questionnaire were fed into the spreadsheet for each single question from 1 -11. An average value was calculated for each question**.** The graph in fig. 2 shows a summary of the results for each single question.

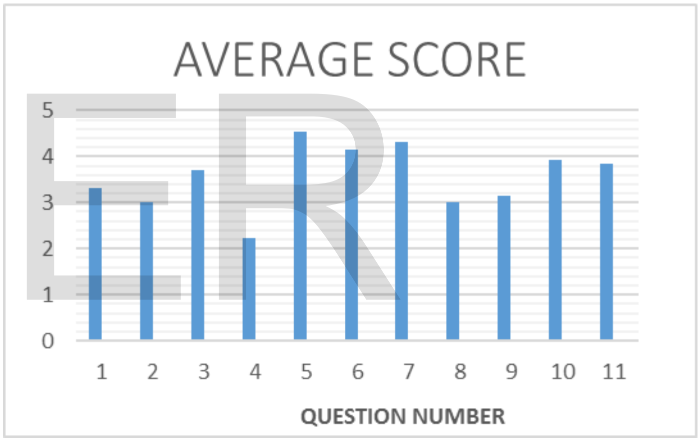


Fig. 2. A graph showing summary for the average score for each of the questions in the questionnaire. The highest possible score is 5 and the lowest possible score is 0. It is obvious that Q5 has the highest average score

2.4 How the Results Were Interpreted

This was based on the average score and the results placed in a table. This was done because the data obtained is quite straightforward to process. The data can be averaged and summarized and results can be interpreted at a glance.

3 Interpretation and Discussion of the Results and Findings

Q1 asks about the familiarity with Agile modelling concepts. This question scored an average of 3.3/5 which implies that the students think that they are quite confident and understand the basic aspects of what Agile is. This might not be the case as it is difficult to quantify familiarity and a short course can just brief them about the idea of Agile. The exposure that comes from a working environment and a number of years in the field cannot be obtained from a study unit of a few months.

Q2 asks if Agile modelling notations are taken from other sources like UML, RUDP, SSADM. Again an average score of 3 is affirmative of this. It implies that the students consider that the notations used in Agile modelling are not purely obtained from Agile only. This is correct in the sense that Agile methods do not place any restrictions on which notations are to be used or not. Agile methods have to be supported by the use of diagrams, but the use of diagrams or notations should be focussed on what is essential to be represented. Even XP (extreme programming) cannot do away with notations. Notations are essential for basic system representation. Obviously Agile methods have to use diagrams from other notations like the UML. But the amount of diagrams used has to be kept to a minimum, because Agile is a lightweight process.

Q3 suggests that too much models create confusion. The average score for this was 3.6/5 which is significant. This means that the students are confident that too much models create confusion. Light weight methods like Agile try to solve this problem. It is true that having too many models do create confusion. It becomes difficult to try to update and keep consistency between many different models.

Q4 negatively asserts that XP (extreme programming) does not require any models at all. This cross checks with Q2 and Q3. The score of 2.23/5 which is relatively low implies that students do not agree with this statement in general. It also shows that the cross check with Q2 and Q3 implies that Q2 and Q3 were not blindly answered. Students in general do not agree with the statement that XP extreme programming does not require any models at all.

Q5 suggests that models do improve software quality by communicating the needs to different stakeholders. This question scored an average of 4.5/5 which is significant. This clearly indicates that models are definitely necessary for Agile methods. It also asserts that students are in agreement that some models for the stakeholders are a must. Students definitely agree that modelling has its own importance and relevance for systems analysis and design.

Q6 asks if students agree with the statement that 'No single modelling approach can work for different problem domains'. Again the score for this question was 4.15/5 which is significant. This implies that the majority of students agree in principle with this assertion. It is a fact that there is no single modelling approach that is suitable for every scenario. This can be seen in the vast modelling techniques that are found in Requirements Engineering and Requirements Elicitation topics. As a matter of fact, for certain systems specific models have to be created. These cannot be used elsewhere because they are specific to a certain problem. Models are limited representation of what happens in the real world. Domain specific models belong to a particular group. Sometimes modelling is more of an art rather than a scientific approach. Throw away prototyping deals with capturing the important elements of a system where there is a great deal of uncertainty. Throw away prototyping is a form of modelling in a certain sense.

Q7 suggests that good models should have a strong visual expression. The score for this question was 4.3/5. This is a significant score indicating that the majority of students understand that good models are important tools for expressing requirements and communicating them properly with the system stakeholders. There are various types of models, some are based on mathematical notations, set theory, algebras, specification languages, etc. These models are not necessary visual ones but they still have their importance. However in literature it is sometimes noted that these structures are not so simple to comprehend by different classes of stakeholders especially those who are not familiar with these notations. On the other hand, many modelling notations like those found in the UML have a strong visual component. This means that they are suitable for expressing requirements with different stakeholders. The UML models can also be formalised using specification languages and things like the OCL (object constraint language) which is part of the UML itself. Hence this question confirms that good models should have a strong visual component. This is evident in many successful notations like those found in FMCs (Fundamental modelling concepts), TAM (Technical Architectural modelling), UML, etc.

Q8 suggests that good modelling can be easily learned in the class room or from books. The average score for this question was 3/5 which is not so significant. This implies that students have different ideas from where modelling can be learned. Some students think that modelling can be learned from the class room whilst some others do not agree with this. In reality modelling basic principles can be learned from a class room environment. However this is not the case with in depth modelling principles. These can only be learned from a proper work environment and learning these principles can only happen after years of hands on experience. This is also seen from Agile methods and Agile modelling where the contributors to these methods are persons who have worked in the software industry for several years and they can contribute to creating these techniques and modifying them.

Q9 suggests that Agile models are not formal representations and hence can contain incorrect views. The average score for this question of 3.15/5 is insignificant. This means that the students in principle agree with this statement to a certain extent. This type of score is to be expected and makes sense. This is because this question is quite tricky and the assertion that agile models are not formal representations and can contain incorrect views is partially correct. Agile models are not formal representations and this is obviously true. However for the business application domain Agile models are normally sufficient to represent the requirements of a system. This implies that even if these models do in fact contain incorrect views nevertheless they are sufficient for business systems that change rapidly in small organizational scenarios. The solution to the problem of inaccurate representation is given in Q10. The solution is itself a principle of Agile where models undergo transformational refinements. This is a key principle of lightweight methods where models are refined as necessary in successive iterations.

Q10 suggests that Agile models require successive refinements. The average score for this question of 3.9/5 indicates that if most of the students are considered, slightly less than 80% agree with the idea of successive iterations for model refinements. This is an important principle for Agile. Agile is an on going process based approach. This signifies that improvements must be done iteratively.

Q11 suggests that Agile models must be validated. The average score for this was quite high at 3.8/5. It is obvious that some form of validation must exist for agile models. The type of validation applicable does not imply that it is complex validation, it could just be some simple form of visual checking or correcting. All models in software engineering require some form of validation whether it is formal or informal. The idea is that when the model is drawn up it has to be checked to see if important parts were omitted and that it makes sense. In reality the model should be verified by a different entity from the one that created the model. In Agile models are also important to test the newly developed applications. This implies that the models must be correct and proper for this important task to be carried out.

4 Final Observations and Conclusions

Here some observations from the case study are presented and discussed. These indicate some core key principles that are an integral part of the so called lightweight methods. Lightweight methods must have some form of planning and structure albeit this can be limited.

The results in the previous sections have been simply interpreted on the average score and experience. The following key observations can be stated and explained.

4.1 Agile methods are based on key principles that are not easily measured and comprehended

Agile is based on ideas of continuous attention to technical quality and detail. This implies that good design will enhance the principle of agility. This is a reason why models require successive enhancements and validation as suggested in Q10 and Q11. Some students assume that Agile is in principle similar to RAD methods. But although Agile might in part resemble RAD efforts for delivery there are intrinsic differences.

Agile is intrinsically different because it is based on quality of design and technical correctness. In Agile simplicity is the key to success. This is the embodiment of light weight methods, where the essence is to simplify and have a clear strong intent and focus on what needs to be done. Simplicity and quality are key principles that are not easily measurable in real terms but they are at the core of Agile method success.

4.2 Understanding the requirements and not just documenting them is an important factor

Diagrammatic notations should serve for comprehension of requirements and not just for documenting the system. This is a very important principle that ties in with Q5 to Q8. The diagrams that are used should serve their purpose properly. The models need to be concise and to the point without excessive detail. Thus a balance needs to be established to get the correct amount of documentation used, which must on one hand not be excessive and on the other hand not too little thus omitting key aspects.

4.3 Attention to excellence in terms of artifacts and design are essential

Agile concepts are heavily dependent on excellence and quality. Agile is a process of self-improvement and learning to do things in the best possible way. This is what excellence means. Excellence implies that the artifacts produced are of the best possible standard. This can only be achieved after years of hands on experience in the field and perfecting the methods of work that are being used. Q10 and Q11 deal with these issues when they state that agile models must undergo successive refinements and that they do require validation. Without proper validation it is impossible to get quality principles. Q9 asserts that agile models are not formal representations hence these may contain incorrect views. This again confirms that it is necessary to have refinements and quality mechanisms in place.

4.4 Agile methods are based on principles of continuous improvement

Agile methods are based on improving the process and the product continuously. This implies that an ongoing organizational mindset is to instill a culture of excellence where continous improvement is the order of the day [7]. Such an approach is not the result of implementing some policy, but it can only be obtained from a companywide approach that has integrated this attitude from a number of years. This idea has not been directly included in the questionnaire, but it is obvious that for having refinements as in Q10 and validation in Q11 the ideal way how to go about this is to have structures that guarantee continuous improvement as part of the organization culture.

References

[1] A. Moran, Agile Risk Management and Scrum, IARM, 2014. http://institute.agileriskmanagement.org/publications/

[2] Managing Agile: Strategy, Implementation, Organisation and People (Springer Verlag, 2015)

[3] A. Moran, Agile Risk Management, SpringerBriefs in Computer Science, Springer-Verlag, 2014.

[4] M. Kunz, R. R. Dumke ; N. Zenker, " Software Metrics for Agile Development", 19th Australian Conference on Software Engineering, ASWEC 2008, IEEE, Perth Wa, Mar 2008, pp. 673-678.

[5] F. Maurer, G. Melnik, "Agile Methods: Crossing the Chasm", Proceeding of the 29th International Conference on Software Engineering (ICSE), IEEE, Minneapolis USA, May 2007, pp. 176-177.

[6] D. Cohen, M. Lindvall, P. Costa, "An Introduction to Agile Methods", Advances in Computers, Vol. 62, Elsevier, 2004, pp. 1-66.

[7] J. Favaro, "Value-Based Management and Agile Methods", Proc. 4th International Conference on Extreme Programming and Agile Processes, Genoa, May 2003.