Formal Experimentation for Agile Formal Methods

Panagiotis Sfetsos[1], Ioannis Stamelos[2]

[1] Department of Information Technology, Technological Education Institute, 54101 Thessaloniki, Greece; e-mail: sfetsos@it.teithe.gr
[2] Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece; e-mail: stamelos@csd.auth.gr

Abstract. The field of knowledge known as Empirical Software Engineering suggests the application of experimental methods in order to discover and describe characteristics of the process and results derived from new methodologies, such as agile formal methods. In this paper we present the methodology needed for conducting formal experiments and we review related work done in the field of the agile formal methods. At the end of the paper, we report on our experimentation work up to now and our suggestions on future work in this important field of research.

1 Introduction

Experimentation provides a systematic, disciplined, quantifiable, and controlled way for evaluating new methodologies and processes [27]. Measurement and evaluation can help to improve the processes and the products in any discipline [12]. Unfortunately, experimentation is not an easy task. The reason is that experiments in software engineering are expensive, time consuming, and difficult to setup. Before the conduction of an experiment, we must first understand in deep the object of the experiment [5]. Then, following exactly the steps needed, as the experimentation methodology suggests, we must organize, prepare and conduct the experiment properly. Choosing the most appropriate methods for data collection and analysis, we have the control of subjects, objects and instrumentation, and the ability to perform statistical analysis using hypothesis testing methods [5], [27]. The methodology needed for conducting a formal experiment will be analytically presented in section 3, describing an experiment having as object an agile method (e.g. XP, eXtreme Programming). Agile methods are gaining increasing recognition as effective methods for building systems in small or medium sized companies and in environments of changing requirements [6], [24]. Nowadays, many practitioners of the agile community are looking for formal methods that may provide fault-free and more effective software development [14], [15], [17], [26]. In rummaging around for such methods, practitioners further claim that formal methods combined with agile methods can guarantee better software development. Some agile practices can take advantage from formal methods such as unit testing, incremental development and refactoring [14], [15]. In order to prove this claim, we must conduct formal experiments comparing the new agile formal method-

Proceedings of the 1st South-East European
Workshop on Formal Methods, SEEFM'03
November 2003, Thessaloniki, Greece

ology with the traditional "heavyweight" methodology. A formal experiment will investigate whether there are any significant differences in the quality, usability, functionality, and robustness of the final products between the two methodologies. In section 4, we review the related literature in experiments related to the field of the agile and agile formal methods and present our experimental work. In section 5, we make suggestions and draw our conclusions in section 6.

2 Background

Before analysing steps being undertaken in our experiment, it is important to mention research work conducted in this area by our predecessors who have devoted much time and effort to ground the scientific task of the experimentation in software engineering. The most significant and commonly quoted researcher, *Victor Basili*, in his historical paper, presented a method about how to collect and analyze data from an experiment [2]. The described method, later called GQM *(Goal-Question-Metric)* still remains one of the basic grounds for research in empirical software engineering. The method consists of six steps, which can be iterated, to describe the process. Basili and his co-authors in a subsequent work [3] provided a complete frame to carry out experiments in software engineering. He defined the experimental process as a four-staged procedure. In their "The TAME Project…" paper [4], the authors provided a set of principles concerning the processes and measurements associated with GQM. Therein, the GQM paradigm and a prototype of an automated tool constructed to support principles and methods were described. In a subsequent paper [5], the authors propose an organisational framework, and use it to combine a set of experiments. They define a method to obtain valid hypotheses from the data collected of a set of experiments. The authors describe again the GQM template and the general procedure to collect data and design experiments. Finally, we must mention the work done by C. Wohlin and his co-author in the book "*Experimentation in Software Engineering - An introduction"* [27], where vivid software engineering experimental process is described in a concrete and concise way.

3 Methodology

The full design of an experiment, from the statistical perspective should include:

- replication of an experiment by using an existing experiment or conducting a new one,
- number of experimental units (teams taking part in the experiment),
- number of blocks (allocated to groups),
- factors (dependent and independent variables),
- number of treatments (for example two: XP and traditional approach),
- levels (categories in every factor), and
- the null hypothesis and the alternative hypothesis.

To perform an experiment, several steps have to be taken and they have to be in a certain order. The steps and the order in which they are executed define the *process* of the experiment. The process may represent a true-experiment or a quasi-experiment (random samples of infeasible participants). The whole experimental process can be divided into the following main activities [3] [27]:

- **definition,** where the problem, the objective and the goals of the experiment are defined.
- **planning,** where the design of the experiment was determined and evaluated. This activity also involves the design of experimental instruments and threats to the conduct of the experiment.
- **operation,** where measurements are collected and analyzed and evaluated in the **analysis** and **interpretation** phase.
- **presentation and package,** where the results are presented and packaged.

3.1 Definition

This section defines the goals of an agile methodology i.e. XP-experiment. The goal template [4], [5], [27] is:

- **analyze.** The application of eXtreme Programming (all practices),
- **for the purpose of** evaluating the XP-discipline compared with the traditional approach,
- **with respect to their** efficiency, reliability, scope, strength, weakness, usability, flexibility and effectiveness,
- **from the point of view of the** researcher- an external observer not engaged with the process,
- **in the context of** i.e Bachelor and|or M.Sc., and|or Ph.D. students as subjects in the experiment.

The context in this eXtreme Programming experiment can be characterized in terms of the number of ***subjects*** (teams) and ***objects*** (projects) involved in the study (see table 1).

Table 1. Experiment context characterization

		# Objects	
		One	**More than one**
# Subjects per Object	**One**	Single object study	Multi-object variation study
	More than one	Multi-test within object study	Blocked subject-object study

It is evident from table 1 that some interesting experiments could be conducted when:

- ✓ every team can carry out several projects, but every project is carried out just by one team (multi-project variation)
- ✓ every team is carrying out just one project, and every project can be attended simultaneously by several teams (multi-test within object study).

3.2 Planning

The definition of the experiment answers the question '*why* the experiment was conducted', while the planning answers the question '*how* the experiment was conducted'. The planning phase of an experiment can be divided into seven steps:

- **Context selection.** Selects the environment in which the experiment will be executed. To achieve the most general results in an experiment, it should be executed in large, real software projects, with professional stuff.
- **Hypothesis formulation.** In the statistical analysis of an experiment, the hypothesis is stated formally and the data collected during the experiment are used with the purpose to reject the hypothesis. If the hypothesis testing under given risks, leads to the rejection of the hypothesis, then real conclusions can be drawn.
  - ✓ **A null hypothesis, H_0.** The use of eXtreme Programming produces as good results as those obtained from the use of traditional approach. This is the hypothesis that the experiment wants to reject. (H_0: $\mu_1 = \mu_2$, where μ_1 express the statistically measured results for the traditional approach, and μ_2 express the statistically measured results for the XP).
  - ✓ **An alternative hypothesis, (H_1, H_2, etc)**. The use of eXtreme Programming produces better results than those obtained from the application of traditional approach. This is the hypothesis in favor of which the null hypothesis is rejected. (H_1: $\mu_1 < \mu_2$, where μ_1 express the statistically measured results for the traditional approach, and μ_2 express the statistically measured results for the XP, thus better results for the XP-practices).
- **Variables selection.** Independent variables that can be controlled and changed in the experiment were selected. The effect of the treatments was measured in the *dependent* variables. Dependent variables could be the time spent in the production process, the quantity of the material generated (size of the product), software assessment by the client (quantitative and qualitative measures).
- **Selection of subjects.** The selection of subjects, statistically called sample from population, is important in experimentation [23]. The sampling of the population can be either a probability sample (when the probability of selecting each subject is known) or a non-probability sample (when the probability of selecting each subject is unknown). Examples of probability sampling techniques are s*imple random sampling, systematic sampling and stratified*

random sampling. Examples of non-probability sampling techniques are *convenience sampling and quota sampling.* In an XP-experiment a randomised block design, where the subjects are randomly divided into groups, could be the proper one.

- **Experimental design. Having examined the hypothesis,** a statistical analysis was chosen and performed to reject the null hypothesis. The general design principles are the following three: *randomization, blocking* and *balancing.*
- **Instrumentation.** In this phase of the experiment instruments were defined for the objects, guidelines (process descriptions, checklists etc.) and measurement instruments concerning data collecting with forms and | or interviews. In an XP-experiment, forms could be used for time and other quantity (i.e. size) measures and interviews for qualitative measures.
- **Validity evaluation.** Validity concerns the results drawn from the experiment. The results have *adequate validity* if they are valid for the sampling population. Four validity categories, related with the methodological approaches are used to eliminate threads in the experimentation, and are presented by T. Cook [7]. These categories are:
  - ✓ ***Conclusion validity*** is concerned with the relation between the treatment and the outcome.
  - ✓ ***Internal validity*** when the treatment causes the outcome.
  - ✓ ***Construct validity*** is concerned with the relation between theory and observation. Are the measurements used the proper ones? For example is LOC/Development time, a proper measure?
  - ✓ ***External validity*** is concerned with the generalization of the conclusions.

3.3 Operation

The collection of data for analysis was carried out during this phase. The operational phase of an experiment consists of three steps:

- **Preparation,** where subjects are chosen and treatments applied on them.
- **Execution,** where the subjects perform their tasks, according to different treatments, and data is collected.
- **data validation,** where the collected data is validated.

3.4 Analysis and Interpretation

Experimental data collected in operation phase must be analysed and interpreted. Quantitative interpretation may be carried out in three steps:

- **descriptive statistics** deals with the presentation and numerical processing of the data (i.e. central tendency, dispersion, etc).
- **data set reduction,** where abnormal or false data points are excluded.

- **hypothesis testing,** where the hypotheses of the experiment are evaluated statistically, at a given level of significance.

3.5 Presentation and Package

This phase embodies the presentation of findings as a research paper, a report for decision-making, and a package for replication of the experiment as educational material.

4 Related Work

There are many published papers focused either on agile formal experimentation or in formal methods. Nevertheless, discussion here will delve on research work done in the area of experimentation in agile methods, by the researchers of Sheffield University F. Macias, M. Holcombe and M. Gheorghe. In their initial paper on the subject, "Experiments of an apprentice of scientist: an empirical assessment of eXtreme Programming", Mascias and Holcombe [19] present a study devoted to evaluate eXtreme Programming through a set of experiments. In a second paper, conducted in 2002 and titled, "Empirical experiments with XP" [20], they present the results of a pilot study, concerning empirical experiments in XP. The researchers mention that a rigorous experiment is underway. In their third paper "Design-led & Design-less: One experiment and two approaches" [21] they presented an experiment comparing two lightweight methodologies.

4.1 Recent Developments

Some important recent developments have been reported during the 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering - XP2003 in Genoa, Italy. Three papers, in the field of formal experimentation and formal methods, were presented. The first one, on pair-programming entitled "Pair-programming effect on Developers Productivity" [13], described an overview of a pair-programming experiment designed to verify how pair-programming affects programmer's technical productivity. The second work, also related to pair-programming, entitled "An Empirical Analysis on the Discontinuous Use of Pair Programming" [16]. The latter showed the efficacy of Pair Programming in transferring knowledge and skills over an environment where people met only occasionally. The third paper was on formal methods entitled "Formal Extreme (and Extremely Formal) Programming" [14]. This paper was an exploratory work where the authors studied how the technology of Formal Methods (FM) could interact with agile process in general and with Extreme Programming (XP) in particular.

4.2 Experimentation Work on Software Engineering

The authors of this paper belong to the Software Engineering group of the Aristotle University of Thessaloniki and often participate in empirical research in a wide ranging Software Engineering fields including; Object-Orientated Analysis and Design , UML, Free-Open Source Software and Agile methodologies especial in eXtreme Programming, to mention but a few. Four formal experiments on Object-Oriented technology and more specifically on Software Maintenance Heuristics and Object-Oriented Design Heuristics were conducted between 2002 and 2003 [10], [8], [9], [11]. A formal experiment was conducted on learning techniques in 2003 [18].

Two formal experiments [25] and [24] were also conducted in 2003. The first, was a controlled experiment, conducted in the Department of Informatics in the Aristotle University of Thessaloniki and investigated learners ability to learning UML Sequence Diagrams, by means of two different approaches (The Traditional learning approach, learning in a classroom with the presence of a teacher and the Distance learning approach using web-based instructional materials). In this experiment, 49 students of the department separated into two groups of 25 and 24 participated.. The second experiment was a replicated controlled experiment, conducted in the Department of Informatics at the Technological Education Institute of Thessaloniki. The initial experiment was conducted by Arisholm [1] and investigated the changeability of a given responsibility-driven (RD) good-design versus an alternative control-oriented "mainframe" (MF) bad-design. In this experiment participated 45 students of the department, separated also in two controlled groups. The results of these two experiments are not published yet.

5 Suggestions

We consider agile formal methods a fundamental means of achieving high quality software. However, there is little understanding of the motivation and empirical hypotheses behind many of these new methods. Due to the lack of research in the agile formal methods area, we are suggesting the creation of an experimentation team in Europe, dealing with such formal experiments. Researchers participating in this team should be experienced in formal experiments and in Agile Formal Methods. Experiments could be conducted in the university environment with our computer science students as subjects. Experiments in different environments, with different subjects, will provide researchers with different and complementary insights into the studied methods. In particular, the members of the Software Engineering group, having such experience, could participate in the organization of this team whose main purpose would be to conduct new or replicate experiments. At this point, we want to present our opinion on the external replicated experiments. We consider the external replicated experiments as important as the new ones. A view that is not widely held is that experiments should be replicated externally to verify and validate original results. As Basili says [4] [5], external replications can benefit from the lessons learned and improve the studied methods, if the constraints and subjects are different.

6 Conclusions

Our initial attempt in this paper was to provide a stepwise-methodology for conducting proper formal experiments in the area of Agile Formal Methods. It is vital that the design, plan and execution of a formal experiment for Agile Formal Method must be suitably and carefully organized, leading to more reliable results and conclusions. A literature review in the field of Agile Formal Methods indicated the lack of experimentation. We suggest the creation of a network of researchers on experiments in Agile Formal Methods (doped as Alliance of Research Engineering Network Team or ARENT) whose sole domain would be to organize formal experiments in this new field of research.

References

1. Arisholm, E, Sjoberg, D, Jorgensen, M. *"Assessing the changeability of two Object Oriented design Alternatives"*. Kluver Academic Publishers, (2001).
2. Basili, V. R, Weiss, D. M. *A methodology for collecting valid software engineering data;* IEEE Transactions on software engineering, vol SE-10, pp. 728-738; Nov, (1984.
3. Basili, V. R, Selby, R. W, Hutchens, D. H. *Experimentation in software engineering;* IEEE Transactions on software engineering, vol. SE-12, pp. 733-743; Jul, (1986).
4. Basili, V. R, Rombach, H. D. *The TAME Project: Towards Improvement-Oriented Software Environments;* IEEE Transactions on software engineering, 14(6):758-773; Jun, (1988).
5. Basili, V. R, Shull, F, Lanubile, F. *Building knowledge through families of experiments;* IEEE Transactions on Software Engineering, 25(4):456-473; Jul-Aug, (1999).
6. Boehm, B., *Get Ready for Agile Methods, with Care.* IEEE Computer, 35(1): p. 64-69, (2002).
7. Cook, T. D, Cambell, D. T. *Quasi-Experimentation-Design and Analysis Issues for field Settings,* Houghton Miffin Company, (1979).
8. Deligiannis, I., Shepperd, M., Webster, S., Roumeliotis M. *"A Review of Experimental Investigations into Object-Oriented Technology."* Empirical Software Engineering Journal 7(3): 193-231 (2002a).
9. Deligiannis, I., Shepperd, M., Stamelos, I., Roumeliotis M., et al. *"An empirical investigation of Object-Oriented Design Heuristics for Maintainability."* Journal of Systems and Software (2002b).
10. Deligiannis, I. Stamelos, L. Angelis, M. Roumeliotis: "The impact of heuristics on Object-Oriented design maintainability: A controlled experiment", *4th International Workshop on Computer Science and Information Technologies CSIT 2002, Rion Patras, Greece,* (2002).
11. Deligiannis I, Stamelos I, Angelis L, Roumeliotis M, Shepperdd M. *"A Controlled Experiment Investigation of an Object-Oriented Design Heuristic for Maintainability"*. Journal of Systems and Software, Elsevier, 2003, to appear.
12. Fenton, N. E, Pfleeger, S. L. *Software Metrics. A rigorous and Practical Approach.* International Thomson Computer Press, 2nd Ed, (1996).
13. Heiberg, S, Salumaa, U, P, Seeba, A. *"Pair-programming effect on Developers Productivity"*. 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering - XP2003 in Genoa, Italy, (2003).
14. Herranz, A, Josι, J, Navarro, M. *"Formal Extreme (and Extremely Formal) Program ming"*. 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering - XP2003 in Genoa, Italy, (2003).

15. Jackson D, Wing, J. *Lightweight formal methods.* Saiedian H et al. An invitation to formal methods. IEEE Computer 1996; 29(4):21–22.
16. Janes, A, Russo, B, Zuliani, P, Succi, G. *"An Empirical Analysis on the Discontinuous Use of Pair Programming".* 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering - XP2003 in Genoa, Italy, (2003).
17. Jones CB. *A rigorous approach to formal methods.* Saiedian H et al. An invitation to formal methods. IEEE Computer 1996; 29(4):20–21.
18. Karoulis, A, Stamelos, I, Angelis, L, Pombortsis, A. "*A Controlled Experiment Investigation on the Impact of an Instructional Tool for Personalized Learning".* Proceedings IEEE Int'l Conference on Advanced Learning Technologies, (2003).
19. Macías, F, Holcombe, M. *"Experiments of an apprentice of scientist: an empirical assessment of eXtreme Programming"* in Memorias del 3er Encuentro Internacional de Ciencias de la Computación (ENC '01); Aguascalientes, México, Set 15-19 / 2001; 875-880.
20. Macías, F, Holcombe, M, Gheorghe, M. *"Empirical experiments with XP"* in Proceedings of 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002), Sardinia, Italy, May 26-30, 2002, 225-228.
21. Macías, F, Holcombe, M, Gheorghe, M. "Design-led & Design-less: *One Experiment and Two Approaches",* Lecture Notes in Computer Science 2675:394-401, (2003).
22. Paulk, M, *"Extreme Programming from a CMM Perspective,"* IEEE Software, Nov.-Dec. 2001, pp. 19-26.
23. Robson, C. Real World Research: *A Resource for Social Scientists and Practioners-Researchers.* Blackwell, (1993).
24. Sfetsos, P, Stamelos, I, *Investigating eXtreme Programming methods application and their link with Newcomer adaptation process in Software Companies.* Department of Informatics in the Aristotle University of Thessaloniki, Internal report (2003).
25. Sfetsos, P, Karoulis, A, Stamelos, I, Angelis, L. *A Controlled Experiment Concerning Traditional and Distance Learning of UML Sequence Diagram.* Department of Informatics in the Aristotle University of Thessaloniki, forthcoming (2003).
26. Uttig, M, and Reeves, S. *Teaching formal methods lite via testing.* Journal of Software Testing Verification and Reliability (STVR), 11(3), p181-195, Sept, (2001).
27. Wohlin, C, Runeson, P. *Experimentation in Software Engineering - An introduction*, Kluwer Academic Publishers, (2000).