Using Configuration Management and Product Line Software Paradigms to Support the Experimentation Process in Software Engineering

Edison Gonzalo Espinosa Gallardo

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software, Universidad Politécnica de Madrid

Madrid, España

egespinosa1@espe.edu.ec

***Abstract*—There is no empirical evidence whatsoever to support most of the beliefs on which software construction is based. We do not yet know the adequacy, limits, qualities, costs and risks of the technologies used to develop software. Experimentation helps to check and convert beliefs and opinions into facts.**

This research is concerned with the replication area. Replication is a key component for gathering empirical evidence on software development that can be used in industry to build better software more efficiently. Replication has not been an easy thing to do in software engineering (SE) because the experimental paradigm applied to software development is still immature. Nowadays, a replication is executed mostly using a traditional replication package. But traditional replication packages do not appear, for some reason, to have been as effective as expected for transferring information among researchers in SE experimentation. The trouble spot appears to be the replication setup, caused by version management problems with materials, instruments, documents, etc. This has proved to be an obstacle to obtaining enough details about the experiment to be able to reproduce it as exactly as possible.

We address the problem of information exchange among experimenters by developing a schema to characterize replications. We will adapt configuration management and product line ideas to support the experimentation process. This will enable researchers to make systematic decisions based on explicit knowledge rather than assumptions about replications. This research will output a replication support web environment. This environment will not only archive but also manage experimental materials flexibly enough to allow both similar and differentiated replications with massive experimental data storage. The platform should be accessible to several research groups working together on the same families of experiments.

I. INTRODUCTION

Albert Einstein said that insanity was doing the same thing over and over again and expecting different results.

Software Engineering (SE) aims to assure that software developers deliver reliable, quality products that satisfy user requirements. On this ground, it should be supported by a process enabling the construction of software systems on time and on budget through the effective and efficient use of human and material resources.

The results of applying a particular software development technology[1] are, today, unpredictable [1]. There is no evidence whatsoever to support most of the beliefs on which software systems construction is based [2]. Nowadays, knowledge of the adequacy, limits, qualities, costs and risks of the technologies used to develop software is missing [3]. Experimentation helps us to verify beliefs and opinions and convert them into facts.

For decades, SE researchers have been running experiments to gather evidence about software development. Outcomes or observations require verification to assure their validity and reliability. It is not possible to draw conclusions from a single experiment. Multiple runs of an experiment are needed to consolidate results, and validate and build the body of knowledge to be disseminated and used in the software industry [4].

According to [5], replication is a key mechanism of the experimental paradigm. Through replication it is possible to verify the results observed in earlier experiments. Campbell and Stanley [6] explain the role of replication as "The experiments we do today, if successful, will need replication and cross-validation at other times under other conditions before they can become an established part of science, before they can be theoretically interpreted with confidence". Judd gives a comprehensible definition of replication [7]: replication means that other researchers try to reproduce the original experiment as closely as possible in other contexts and using different samples. If the results of the replication are consistent with the original research, this raises confidence in the hypothesis supported by the original study.

According to [8] and [9], replication plays a major role in empirical software engineering (ESE), enabling the research community to build knowledge about the outcomes or observations under set conditions.

---

1 We use the term technology to refer to software development methods, techniques, methodologies and tools.

Replication has not been an easy thing to do in SE because the experimental paradigm applied to software development is still immature. The main reason for the shortage of replications in ESE is that it is hard to describe an experiment in enough detail for another researcher to be able to replicate it accurately [10]. The context of an SE experiment is extremely complex because the phenomenon under study has very many variables, and software development (and, therefore, SE experiments) involves human beings.

Some researchers argue that having an experimental package provided support for their experiment replication process. A traditional experimental package is a package composed of a set of structured processes, documents, materials, etc., used by experimenters to replicate an experiment. A replication package is then the means of communication and coordination among the authors of the experiment and experimenters in replicating the experiment [11], [12]. A full replication package includes all the tasks, checklists, and procedure and problem-solving descriptions, etc., requested from the package authors [13]. Basili [12] suggests that, if there is a replication package, experimenters should use it and, if not, they should be build one.

Despite the fact that there are several ESE replication packages, the truth is that there are still very few replications in this field. For some reason, experimenters are not finding replication packages as helpful as expected. This suggests that the solution offered so far to the SE replication problem is naïve [14]. The crux of the matter is that there are many factors that can alter experimental results, and they are hard to predict beforehand [10].

Like SE, other experimental disciplines have similar trouble controlling the experimental context. To successfully replicate ESE experiments, it is necessary to move away from the concept of replication in the natural sciences [15], where experiments can be reproduced exactly. ESE researchers need to look at approaches in other less strict experimental disciplines that have not traditionally been a source of inspiration for ESE. These disciplines consider replication as the repetition of an experiment by other researchers in other environments with other samples in an attempt to reproduce research as closely as possible [7]. This implies accepting that replications are pretty accurate adaptations and not necessarily exact copies of the original experiment.

Gomez [5] have found that the elements and structure of the experiment to be replicated tend to vary depending on how the replication is run. Between-replication variation is useful for different verification purposes.

The approach that we propose to help to solve the replication problem is to analyse and understand the potential of non-exact (differentiated) replications in accordance with a concept of replication that is closer to ESE reality. This should be a dynamic process that supports the replication process. First, it should help to transfer information to experimenters so that they know how to correctly apply instruments and materials. Second, it should provide an infrastructure for storing information about components (materials, instruments, etc.) and identified replications in order to guarantee their integrity, reliability and traceability for reuse within the experimental research cycle.

II. Problem Statement

Replication has not been an easy thing to do in software engineering (SE) because the experimental paradigm applied to software development is still immature.

ESE researchers have proposed several instruments for transferring information among experimenters to improve the performance of replications in SE. In early replications run in SE, publications reporting the experiment were the only documentation transferred about the original experiment. Some researchers proposed static replication packages (repository containing materials) including some documents required to replicate an experiment (documentation to be delivered to subjects, data collection forms, etc.) in an attempt to improve, ease and incentivize replications. More details are required than just a description of and the provision of the materials required to execute the experiment in order to replicate an ESE experiment. Other things added to the replication package were data collected during the experiment and materials required to train subjects, as well as experiment execution procedures (for example, a script with tasks to be performed by the experimenter during the experiment operation).

Some examples of the many papers focusing on experimentation and replication in SE are (in chronological order): [11] and [16], creating a laboratory kit (i.e., a package containing all the experimental materials, data, and analysis) to facilitate replication; [17], referring to the creation of an experimental package to support external replications; [18], [12] and [19], suggesting that it can take many months to prepare an experiment that can be run in no time at all; [20], using the experimental package built by [21], [4], discussing tacit knowledge and replication packages; [22], [23], [24] and [25], which are studies mentioning or suggesting the creation and use of replication packages.

ESE researchers have tried to verify experimental findings by repeating experiments using a traditional replication package, which would appear, for some reason, not to have been overly helpful for running replications in this field. The cause would appear to be some of the package-related problems detailed below.

- A traditional replication package is a static container (containing documentary information) that hosts different versions of materials modified and/or created as a result of variations of replications to be run or by adopting experimental paradigm processes in SE. This has increased the quantity of package materials, leading to two problems. The first concerns management to control the integrity and traceability of the different versions of the materials to be used to generate different replications. The second is that it is harder for experimenters to handle the materials.
- Replications are difficult to configure and manage using the package (different versions of package

materials are required in similar or differentiated replications to execute the experiment).

- The costs of preparing multiple processes and materials (artefacts, components) contained in the package and used in the experimentation processes increase considerably.
- Iterative improvement of materials content for use in replication of experiments.
- There is limited access to pragmatic information on the success or failure of using the replication package, unless experimenters have used it before. Researchers do not share what they learn from using a package with others. Thus they miss out on the chance of learning from other people's experiences.

The traditional packages used for replication play a key role in experimental research, but they have not solved the experimental replication process problem. There are still very few replications in SE.

III. Proposed Methodology

We will undertake this research by enacting generative cycles to build a preliminary contents structure by successive approximations, where each approximation accounts for a different viewpoint. The first approximation considers existing theoretical knowledge, and the next adds knowledge elicited from researchers experienced in experimentation. The preliminary structure then has to be inspected by experts. This will presumably lead to it being refined. Finally, the reviewed structure will undergo observation and later experimentation. The researcher will observe the structure and check aspects like feasibility (capability to represent replications) and/or flexibility (capability to represent all replication types). The experiments will be run with potential contents structure users. The input will be documents generated by a series of experiments executed by the GRISE research group in the testing field. These documents will be put together by applying elicitation techniques, like interviews, observation and case analysis, and by running experiments.

This research is at the first stage. We have analyzed the documents generated from experiments and examined the software configuration management (SCM) process to find out whether this paradigm is applicable in experimentation. SCM is a discipline applying a set of procedures to control and maintain both the integrity and traceability of software development process products. Considering replication packages to be frequently changing products that we want to trace throughout the research cycle, we developed an analogy considering key SCM concepts and activities. The research results were:

- Document called Adaptation of the SCM Process to the Experimental Research Cycle reporting the analysis of SCM concepts and activities for adaptation to the ER cycle.
- Document called Experiment Configuration Management Plan (ECMP) containing a set of activities and processes to be used to manage experiments.
- Instantiation of a case applying ECMP.

We are now studying the software product line (SPL) paradigm to analyze the feasibility of applying it to the experimental replications configuration process. First, we applied feature-oriented domain analysis (FODA) to the gathered digital documents in order to abstract and establish common features and variants at the materials level (objects, instruments, documents, etc.), which we then used in different experiment runs. Second, we have built both a matrix and a hierarchical tree showing the identified common characteristics and variants, which are now under review.

IV. Progress / Results

Thus far, progress has been made concerning research into hitherto the following topics:

- Study of the concept of what the social sciences call differentiated replications for application in and adaptation to ESE.

  Study of the feasibility of adapting the software configuration management (SCM) process to the experimental research cycle. In the context of ISO/IEC [26], the software life cycle is viewed as a reference framework containing the processes, activities and tasks involved in software product management, development, use and maintenance. According to [27], the software system life cycle establishes a succession of stages through which a software product passes, starting when the product is conceived and ending when the software is obsolete. Another characteristic is that it specifies a pre-established order of stages and establishes criteria for moving from one stage to another. This specified and defined process framework includes a set of processes divided into three groups: basic, for support and organizational processes. SCM is one of the support processes.

  SCM process products, called configuration elements (CE), are created at each stage of the software development life cycle. Changes to the products can occur at any time as a result of the interaction among people participating in the different software project stages or the development, use and maintenance of the actual software features. According to [28], the system will change irrespective of where we are in the system life cycle, and the drive for change will continue throughout the life cycle.

  We refer to the set of CEs produced, used, modified and exposed to formal and informal configuration processes in a software project as SCM.

  According to [29], SCM is the art of coordinating software development to minimize confusion. SCM is the skill of identifying, organizing and controlling changes to software built by a team of programmers.

SCM is then a formal engineering discipline that is part of system configuration management and provides methods and tools for identifying and controlling software throughout its development, use and maintenance.

Experimentation is a process that is divided into several phases. Each phase outputs experimental products (materials, instruments, objects, etc.) that are part of the SE experiment. Experimental products could be considered as a reusable common set of interrelated products stored in repositories that are used to enact future experimentation processes in the experimental research cycle. Different versions of products, generated by changes made to different levels of the experiment, such as the experimental domain, the experimental subject matter, the problem or elements like protocol or design, etc., are required for further replications. The research run establishes the minimum contents required to deploy the proposed experimental elements configuration management (EECM) process for application throughout the experimental research cycle. The EECM process affords means for guaranteeing the integrity, reliability, consistency and traceability of experimental research cycle experimental products. The experimental products will be selected, identified, recorded and controlled during the experimentation, replication and synthesis processes.

- Study for applying software product lining in the experimental research cycle.

According to [30], a software product line (SPL) is a set or family of products that share a set of features that satisfy some special market needs and have been developed from a set of core assets with a particular production plan.

This definition includes two key concepts for applying SPLs: features and core assets. A feature is a conceptual characteristic of a system and is used to describe or distinguish a product of a line. A core asset is a software artefact that is used to produce one or more products of a software product line. It can be a software component, a process model, an architecture, a document or any other system development process output.

Software product lining has proved to be an effective means of taking advantage of code reuse. It leads to efficient development, a short time to market and quality products [31]. In SPL, a related set of products is developed by combining reusable core assets with specific products (custom assets). Core assets implement most of the product functionality and support variable functionality (reference architecture). Custom assets are built into core asset products. They instantiate changes and implement unique product functionalities.

In order to apply SPL in SE experimentation, we will analyse information from research areas in which the group is experienced: software validation, requirements and usability. This information will be gathered by applying elicitation techniques, like interviews, observation and case analysis, and by running experiments. We will research, analyse and select techniques for abstracting and establishing common features (reusable) and variants at artefact (product) level that are typical across different experimental runs. Elicitation sessions with experts in the experimentation process (people with different profiles) will be necessary to establish (by different means) the information that they consider important for experimental replication. A structure of common contents and variants (products) will be useful for configuring different types of replications by selecting or rejecting set features.

V. Related Work

Very active research lines in ESE are: improvement of experiment reporting by generating guidelines, such as [32], [33] or [34]; enrichment of quantitative SE experiments using qualitative experimental methods along the lines suggested by [35], which is being pursued in Spain by the Kybele group [36], or experimentation at the clinical level (to use an analogy with experimental medicine), rather than in the laboratory, getting industry to carry out their own experimental studies, as the Fraunhofer Institute for Software Engineering or Microsoft [37] are doing. But this project is part of a line of research on the generation of evidence by combining the outcomes of more than one experiment.

The aim of research on replication is to better understand the concept of replication for adaptation to SE reality. This is what researchers at the University of Strathclyde, [14] and [15], are working on. Our group has contributed to this line [38]. Another aim is to improve the transmission of information using replication packages. Basili [39] pioneered this research and was later joined by Conradi [40] and Travassos [41]. Our group is also working on this line [10].

The research planned as part of this project also touches upon a more technological line of ESE research involving the construction of experimentation support environments. These environments can range from a simple experimental materials repository to the definition and implementation of tools for use in experimental process activities. Examples of such environments are SESE [42] eSEE [43] and Ginger2 [44]. There have also been attempts at storing results (of experiments, case studies and even singular experiences) in what we might call experience repositories, like cebase [45] and Visek [46].

This project is part of a line of basic research on evidence generation, and especially replication and aggregation, plus a line of applied research into technology to support the two tasks.

REFERENCES

[1] C. Wohlin, P. Runeso, M. Höst, M. Ohlsson, B. Regnel and A. Wesslén, Experimentation in SE: An Introduction, 2000.

[2] N. Juristo and A. Moreno, "Reliable knowledge for software development," *IEEE Software,* 2002b.

[3] Jedlitschka and Ciolkowski, "Towards evidence in SE," Proc. of ACM/IEEE Int. Symp. on Empirical SE, 2004.

[4] S. Forrest, V. Basili, J. Carver and J. C. Maldonado, "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem," Proceedings. 2002 International Symposium, pp. 7-16, 2002.

[5] O. S. Gomez, N. Juristo and S. Vegas, Replications Types in Experimental Disciplines, 2010.

[6] D. T. Stanley and C. Campbell, Experimental and Quasi-Experimental Designs for Research., Hope-well, NJ: Houghton Mifflin Company, 1963.

[7] Judd, Smith and Kidder, Research Methods in Social Relations, Jovanovich College Publishers, 1991.

[8] J. Forrest, C. Shull, J. Carver, S. Vegas and N. Juristo, "The role of replications in Empirical Software Engineering," Springer, pp. 211-218, 2008.

[9] C. Knutson, J. Krein, N. Juristo and L. Prechelt, "1st International Workshop on Replication in Empirical Software Engineering Research (RESER)," ICSE, 2010.

[10] S. Vegas, N. Juristo, A. Moreno, M. Solari and Letelier, "Analysis of the influence of communication between researchers on experiment replication," Proc. of the ACM/IEEE Int. Symp. on Empirical SE, 2006.

[11] E. Kamsties and C. Lott, "An empirical evaluation of three defect detection techniques," Technical Report ISERN 95-02, May 1995.

[12] V. R. Basili, F. Shull and L. Filippo, "Building Knowledge through Families of Experiments," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 25, no. 456-473, 1999.

[13] A. Stefik, S. Siebert, S. Melissa and K. Slattery, "An Empirical Comparison of the Accuracy Rates of Novices using the Quorum, Perl, and Randomo Programming Languages," 2011.

[14] Brooks, Roper, Wood, Daly and Miller, Replication's Role in SE En Shull, Singer, Sjoberg (Eds.) Guide to Advanced Empirical SE, Springer-Science, 2007.

[15] Miller, "Replicating SE experiments: A poisoned chalice or the holy grail," Information and Software Technology, 2005.

[16] A. Porter, L. Votta and V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 21, no. 6, 1995.

[17] J. Daly, A. Brooks, J. Miller, M. Roper and M. Wood, "An Empirical Study Evaluating Depth of Inheritance on the Maintainability of ObjectOriented Software," ISERN, 1996.

[18] P. Fusaro, F. Lanubile and G. Visaggio, "A Replicated Experiment to Asses Requirement Inspection Tecnique," Kluwer Academic Publishers, 1997.

[19] A. Brooks, M. Roper, M. Wood, J. Daly and M. James, "Replication of Software Engineering Experiments," 2000.

[20] M. Wood, R. Marc, A. Brooks y J. Miller, «Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study,» 2003.

[21] E. Kamsties and C. Lott, "An Empirical Evaluation of Three Defect-Detection Techniques," Proceedings of the Fifth European Software Engineering Conference, 1995.

[22] J. Natalia and S. Vegas, "Functional Testing, Structural Testing and Code Reading: What Fault Type do they Each Detect," Springer, p. 208–232, 2003.

[23] D. I. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg y A. C. Rekdal, «A Survey of Controlled Experiments in Software Engineering,» IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 31, 2005.

[24] S. Vegas, «Maduración de Conocimiento Mediante una Familia de Experimentos,» 2005.

[25] C. Apa, M. Solari, D. Vallespir y S. Vegas, «Construcción de un paquete de laboratorio para un experimento en Ingenieía en Software,» 2011.

[26] IEEE/EIA 12207, Standard for Information Technology Software life cycle processes Implementation considerations., 1997.

[27] IEEE Std 610.12, Standard Glossary of Software Engineering Terminology, 1990.

[28] Bersoftf, Henderson and Siegel, Software Configuration Management, Prentice Hall, 1980.

[29] W. Babich, Software Configuration Management, Coordination for Team Productivity., 1st edition. Boston: Addison-Wesley, 1986.

[30] A. Wesley, Software product lines: practices and patterns, Boston, 2002.

[31] P. Clements and C. W. Krueger, "Being Proactive Pais Off/ Eliminating the Adoption Barrir.," Point Counterpoint article in IEEE Software, Julio/August 2002.

[32] Singer, «Using the American Psychological Association (APA) Style Guidelines to Report Experimental Results,» Proc. of the IEEE Workshop on Empirical Studies in Software Maintenance, 1999.

[33] Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, E. Emam y Rosenberg, «Preliminary Guidelines for Empirical Research in SE,» IEEE Transactions on SE, vol. 28, nº 8, 2002.

[34] P. Jedlitschka, «Reporting Experiments in Software Engineering,» Proc. of the ACM/IEEE Int. Symp. on Empirical SE, 2005.

[35] Seaman, «Qualitative Methods in Empirical Studies of Software Engineering,» IEEE Trans. on SE , vol. 25, 1999.

[36] M. Lázaro, «Experiences in Integrating Qualitative and Quantitative Methods,» Proc. of the ACM/IEEE Int. Symp. on Empirical SE, 2006.

[37] Nagappan, «Empirical case studies in industry: Some thoughts,» Empirical SE Journal, vol. 132, 2007.

[38] Gómez, Juristo and Vegas, "Replications Types in Experimental Disciplines," Proc. of the 4th ISESE15, 2010a.

[39] Basili, Green, Laitenberger, L. Shull, Sorumgaard and Zelkowitz, "Packaging researcher experience to assist replication of experiments," ISERN Meeting, 1996.

[40] Conradi, Basili, Carver, Shull and Travassos, "A Pragmatic Documents Standard for an Experience Library: Roles, Documents, Contents and Structure," University of Maryland CS-TR-4235, 2001.

[41] Shull, Basili, Carver, Maldonado, Travassos, Mendonca and Fabbri, "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem," Proc. of the ACM/IEEE Int. Symp. on Empirical SE, 2002.

[42] Arisholm, Sjøberg, Carelius and Lindsjørn, "SESE -- an Experiment Support Environment for Evaluating Software Engineering Technologies," Proc. of the 10th Nordic Wkshp on Programming & Sw Dvlopmnt Tools and Techniques., 2002.

[43] Lopes and Travassos, Knowledge Repository Structure of an Experimental Software Engineering Environment Proc. of the XXIII Brazilian Symposium on Software Engineering., 2009.

[44] Torii, Matsumoto, Nakakoji, Takada, Takada and Shima, "Ginger2: Computer-aided," ESE TSE15, vol. 25, no. 4, 1999.

[45] Boehm and Basili, The CeBASE Framework for Strategic Software Development and Evolution Proc. of the 3rd International Workshop on Economics-Driven Software Engineering Research., 2001.

[46] Hofmann and Wulf, Building Communities among Software Engineers: The ViSEK Approach to Intra- and Inter-Organizational Learning Lecture Notes in CS 2640., 2003.

[47] IEEE 730, IEEE Standard for Software Quality Assurance Plans, 2002.

[48] V. Basili y S. W., «Comparing the Effectiveness of Software Testing Strategies,» University of Maryland. Technical Report TR-1501, May 1985.

[49] M. C. Bastarrica, «Arquitectura base en una línea de productos de software,» 2002.

[50] N. Juristo, A. Moreno and S. Vegas, "TOWARDS BUILDING A SOLID EMPIRICAL BODY OF KNOWLEDGE IN TESTING TECHNIQUES," 2004.

[51] N. Juristo y A. Moreno, Basics of Software Engineering Experimentation, 2001.

[52] M. Staples, «Change Control for Product Line Software Engineering,» 2004.

[53] ANSI/IEEE Std 828 2005, Standard for Software Configuration Management Plan. IEEE. 2005, 2005.