

Modeling the Experimental Software Engineering Process

Miguel Goulão, Fernando Brito e Abreu
FCT/UNL, CITI, QUASAR research group
{miguel.goulao|fba}@di.fct.unl.pt

Abstract – Systematic reviews on software engineering literature have shown an insufficient experimental validation of claims, when compared to the standard practice in other well-established sciences. Poor validation of software engineering claims increases the risks of introducing changes in the software process of an organization, as the potential benefits assessment is based on hype, rather than on facts. The software engineering community lacks widely disseminated experimental best practices. This paper contributes with a model of the experimental software engineering process that captures the best practices in the area and is aligned with recent proposals for experimental data dissemination. This process model can be used either as a support in the definition of software engineering experiments or in conducting comparisons among experiment results.

Index terms – Experimental software engineering, process modeling, scientific method

I. INTRODUCTION

The scientific method (aka experimental process) is used in mature sciences, such as physics, chemistry, or biology, in the process of trying to explain phenomena. Through the collection of observable, measurable information, evidence is gathered to help hypothesis refusal or acceptance. Without such an approach, conclusions are subject to a potential bias introduced by the researcher. The experimental process is expected to be extensively and unambiguously documented, to facilitate its scrutiny and independent replication by peers.

The state of practice in computer science and software engineering is different. A survey has reported that, out of 5453 scientific articles published in 12 leading software engineering journals and conferences from 1993 to 2002, only 103 (1.9%) of them reported controlled experiments in the realm of software engineering tasks [1]. In the same survey only 14 series of experiment replications were found, of which just 6 were performed independently (not by the original authors). Five series included replications that, at least partially, rejected the findings of the original experiment. Only one of the latter rejections was reported by the original team.

The low percentage of experiment-based validation in papers (less than 10%), when compared to other research methods is noticeable, both in the context of software

engineering [2] and computer science [3]. According to [4], the percentage of published papers in computer science that make claims that should require experimental validation support, but provide none is around 40% for computer science papers, and 50% for papers on software engineering. In other scientific areas, such as optical engineering, physics, psychology or anthropology, only around 15% of the papers present no experimental validation. The insufficient validation of claims has also been reported in [5].

Although in some cases a theoretical validation of claims is more adequate than an empirical one, our ability to assess improvements achieved with the use of a new tool, language, process or technique, is often hampered by the lack of sound empirical evidence. Expert's opinions are mostly based on personal experience and intuition, therefore biased by the expert's background and are more vulnerable to hype.

Typical justifications for not performing experimentation include excessive costs, uselessness, and the difficulty in performing experiments. Authors such as Tichy have argued against this kind of excuses [6]. Both industry and academia should encourage evidence-based scrutiny, rather than dismissing it. Promoting best practices has been recognized as a way to help surpass adoption skepticism. Besides providing evidence on the feasibility and usefulness of experimentation, best practices may help to mitigate implementations costs (best solutions are usually more cost-effective).

In this paper, we model the experimental software engineering (ESE) process, capturing current experimental best practices. Our model is aligned with recent proposals on experimental results reporting guidelines, aimed at facilitating results comparability. Such ability is an essential building block for the creation of a software development body of knowledge supported by sound empirical evidence. Our model can be used as a guideline for practitioners involved in adopting experimental practices to leverage software process improvement efforts in their organizations. It may also be useful for researchers interested in building and disseminating sound empirical evidence on their scientific claims.

This paper is organized as follows. Section 2 discusses the concept of evidence-based software engineering and how it can impact current software development practices. Section 3 presents the experimental software process model. Section 4 frames this model into current related research efforts. Finally,

section 5 discusses the main conclusions of the work presented in this paper.

II. EVIDENCE-BASED SOFTWARE ENGINEERING

A. *The benefits of evidence*

Evidence-based software engineering (EBSE) is a paradigm that supports arguments concerning the suitability, limits, costs and risks, inherent to software engineering tools and techniques, with experimental evidence. The goal of EBSE is to provide the means by which current best practices from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software [7, 8]. Fagan inspections [9, 10] are among the most notable and pioneer targets of successful EBSE. Its traditional model encompasses a joint meeting of a moderator, a reader, a recorder and the author of the inspected deliverable, where identified defects are merged and recorded. However, experimental assessment has shown that there is no significant difference on the number and kind of defects found with meetingless Fagan inspections [11], therefore justifying this more cost-effective alternative. Other examples include establishing correlations between OO design metrics and software maintainability [12] and CASE tools benchmarking [13].

If we regard the evolution of the software engineering body of knowledge as a quality driven process, the evidence-based approach fits well into it. Both Deming's plan-do-check-act cycle [14] in management, and Basili's Quality Improvement Paradigm [15], for software production, rely on the objective analysis of data collected in projects to detect opportunities for improving the processes under analysis. Following this perspective, software engineering proposals should be assessed for soundness. Experimental assessment would be an adequate technique to measure the extent to which new proposals improve existing practices. This is also in line with the requirements of the highest levels of maturity in software development, according to maturity models such as CMMI [16] and OOSPICE [17].

B. *The pitfalls of evidence*

In general, a single experiment cannot be expected to provide definitive evidence on a phenomenon. Even carefully planned single experiments are subject to threats to validity that may bias their conclusions. Experiments replication can mitigate such threats, particularly by introducing variations to the experimental design. While these replications, also known as **differentiated replications**, may have weaknesses of their own, the evidence collected in several replications can be used to confirm, or refute, the conclusions of the original experiment.

In contrast, **close replications** try to maintain as much as possible the conditions of the replicated experiment. They are useful both as confirmatory studies and as a means of uncovering previously unnoticed sources of variation in an experiment.

The aggregation of a body of knowledge from the collection of lessons learned in disperse independent research

efforts (experiment replicas) remains an unsolved challenge within the software engineering community. A narrative review of those replications is helpful to understand their results, but may not be adequate to draw conclusions on their most controversial aspects, due to the potential bias caused by the reviewer's beliefs. To combine the results of replicated studies we need an approach that accounts for the different strengths and weaknesses of those studies, such as their sample size or the type of statistical tests applied. Brooks advocated the use of meta-analysis for this purpose [18]. Miller attempted to perform a meta-analysis on a set of independent defect detection experiments, but found serious difficulties concerning the diversity of the experiments and heterogeneity of their data sets, therefore being unable to derive a consistent view on the overall results [19]. The development of collaborative efforts for the creation of common repositories for ESE studies, inspired by examples such as that of the Cochrane Collaboration (www.cochrane.org), dedicated to health care evidence data collection, is a long-term goal of the software engineering community. An attempt to bootstrap such a repository was made within the ESERNET European initiative, where several research efforts were coordinated with the intent of gathering a large number of comparable studies [20]. Unfortunately, along with a fairly high mortality of experiments, typically justified by funding problems, the studies were too scattered and too diverse to allow meaningful comparison. One of the conclusions of this initiative was that less expensive studies, such as post-mortems on existing data, would be useful, and more feasible, although they have a lower validity than controlled experiments and quasi-experiments [21].

C. *Replication difficulties*

A possible explanation for the lack of experimentation in software engineering may be that researchers are not so keen to perform replications due to their apparent lack of novelty. However, the software engineering community encourages replication in conference series such as METRICS and ISESE (merged in ESEM on 2007), or journals such as the Empirical Software Engineering (Springer).

A deeper obstacle to replication may be its inherent difficulty. Even with experiment packages prepared by veteran experimenters to their peers, the tacit knowledge problem remains an obstacle. This problem occurs due to implicit knowledge that is not sufficiently documented when preparing experiment packages [22]. The existence of a model for experimentation, along with reporting guidelines, makes the information needs in experiment packages more visible, facilitating replication activities. The publication of guidelines for empirical software engineering research in general [23-25], systematic reviews [7], and controlled experiments [26, 27] has received some attention over the last few years. While none of these proposals has yet emerged as a community-wide standard, they do convey a growing concern for the comparability among different experiments, through the harmonization of the experimental reports.

III. A MODEL FOR THE ESE PROCESS

In this section, we present a process model for experimentation in software engineering. Its deliverables are mapped into an underlying logical model of experiment-related concepts that covers the information needs of the experiments reporting guidelines proposed in [27].

Figure 1 presents an overview of the process. Its activities will be discussed in the next sub-sections.



FIGURE 1 - OVERVIEW OF THE EXPERIMENTAL PROCESS

A. Requirements definition

From a process viewpoint, the first step is to clearly state the research problem one is trying to address. The context of the experiment should be defined, as it will constrain, along with the research problem, the definition of the objectives of the experiment. On the other hand, the objectives of the experiment also influence the options made by experimenters, in what concerns the context, hence the feedback loop between these two activities (Figure 2). One might expect the experiment's objectives to come first, followed by a context definition to support those objectives. As experiments are typically carried out in a resource-constrained environment, the objectives of the experiments have to be adjusted to the contexts available to experimenters.

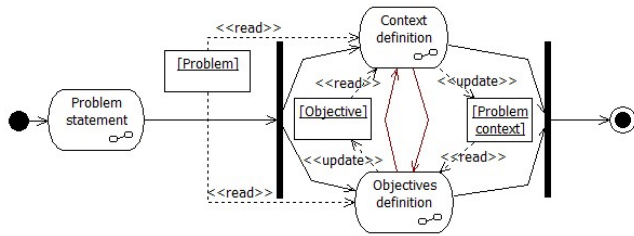


FIGURE 2 - EXPERIMENT REQUIREMENTS DEFINITION

A.1. Problem statement. Software engineering is a problem-solving discipline. One should start by clearly defining the problem that the experiment will address, as well as identifying where it can be observed, and by whom. It is important to state how solving the identified problem is expected to impact on those observing it (Figure 3).

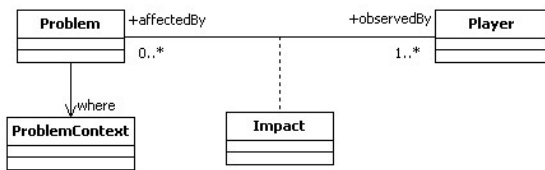


FIGURE 3 - PROBLEM STATEMENT

A.2. Context definition. The experiment context determines our ability to generalize from the experimental results. Experiments can be conducted in different contexts, each with its own benefits, costs, and risks. These constraints have to be made explicit, in order to ensure the comparability among

different studies, and to allow practitioners to evaluate the extent to which the results obtained in a study, or set of studies, are applicable to their particular needs. At this phase in the process, an informal assessment of the context is sufficient. This topic is revisited during the experiment design phase, where a more systematic characterization of the context should be performed.

A.3. Objectives definition. One should clearly define the experiments' goals. Building upon Basili's earlier work [28], Wohlin *et al.* proposed a framework to guide the experiment definition [24]. The framework is to be mapped into a template with the following elements: the **object of study** under analysis, the **purpose** of the experiment, its **quality focus**, the **perspective** from which the experiment results are being interpreted, and the **context** under which the experiment is run (Figure 4).

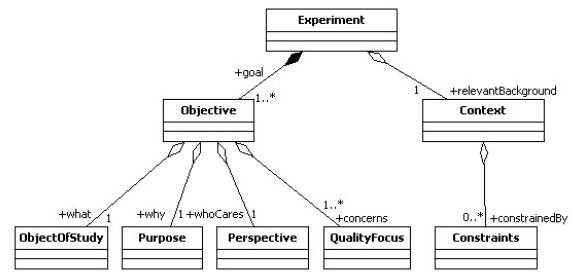


FIGURE 4 - EXPERIMENTAL OBJECTIVES AND ITS CONTEXT

The framework is interesting both from the point of view of someone performing the experiment and that of someone using the experiment's results. The former makes explicit a goal-driven approach to his experimental work. This exercise helps delimiting the experiment's boundaries and focusing on its essential goals. The latter accesses a systematic description of experiment goals, which facilitates the assessment of the experiment's relevance to his context.

B. Design planning

While the experiment definition was about why a particular experiment is performed, the experiment design planning is about how it will be performed. The context of the experiment is revisited here, with more detail. Figure 6 describes the activities related to the definition of the experiment design that will be discussed in detail in the subsequent sections.

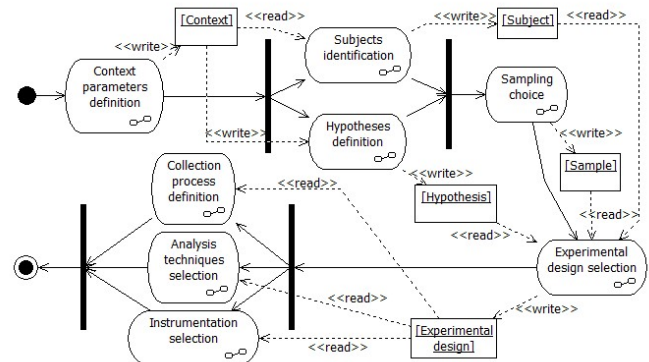


FIGURE 5 - EXPERIMENT DESIGN PLANNING

B.1. Context parameters definition. Throughout the experiment several context parameters remain stable (their value is the same for all the subjects). Therefore, we can safely assume differences observed in the results cannot be attributed to these parameters. While the actual set of parameters to be reported may vary, a core set of context parameters was identified in [24], as depicted in Figure 6.

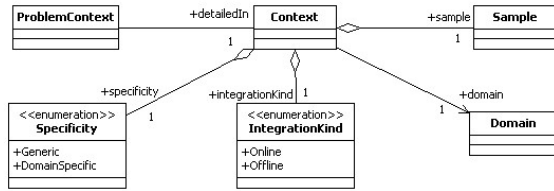


FIGURE 6 - CORE EXPERIMENT CONTEXT PARAMETERS

Concerning their integration within the development process, experiments can be conducted either **on line**, or **off line**. The former, carried as part of the software process in a professional environment, involves an element of risk, since they may become intrusive in the underlying development activity.

An orthogonal classification of context concerns the people involved in the experiment. One may choose among performing the experiment with professional **practitioners**, or with **surrogates** for those practitioners (e.g. students). The first option leads to results that are more easily comparable to others obtained in a professional context, but care must be taken to reduce potential overheads to practitioners activities. Using students as surrogates for professional practitioners is less expensive, but makes the experimental results harder to extrapolate for a professional community. To reduce the gap to practitioners the researcher should prefer using graduate students.

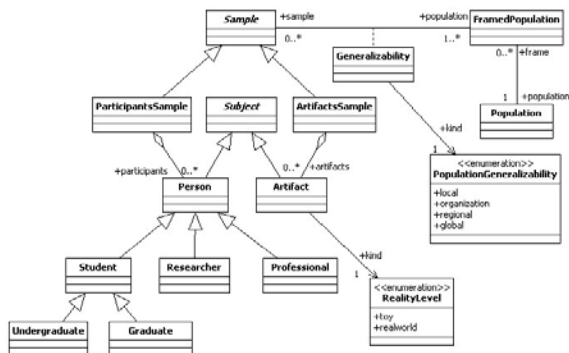


FIGURE 7 - SAMPLE CHARACTERISTICS

Another dimension constraining the experiment is the usage of **toy** vs. **real world** problems. The scarce resources available for the experiment (e.g. limited time subjects can devote to the experiment) and the risks concerned with its outcome often constrain problem selection. The latter relates to the potential harm caused by the outcome of the experiment (e.g. while experimenting with different testing techniques on a real problem, a less effective technique being tested could lead to a lower final product quality being delivered to a customer). Toy problems are often used in early experiments, due to lower cost and risk. If a toy experiment outcome is

satisfactory, the risk of scaling up to a real problem is mitigated.

Experiments can also range from **specific** to **general** in the sense that their results are applicable to a niche or to a wider population. For instance, when experimenting with the maintainability of object-oriented software, one can design experiments that are language-specific, or experiments that yield results applicable to object-oriented software in general.

Other relevant parameters can be added to this core set. For instance in [25] it is argued that context information such as the software application domain or organizational constraints such as the development process used by subjects performing the experiment should also be included.

B.2. Hypothesis formulation. This formulation should be stated as clearly as possible, and presented in the context of the theoretical background it is derived from. This theoretical context makes the hypothesis' implications more apparent, and is important to facilitate the inclusion of the experiment's outcome in the software engineering body of knowledge [25].

Figure 8 models the main concepts involved in hypothesis definition and the goals driving that definition, along with the basic concepts concerning variables selection, discussed in the next section.

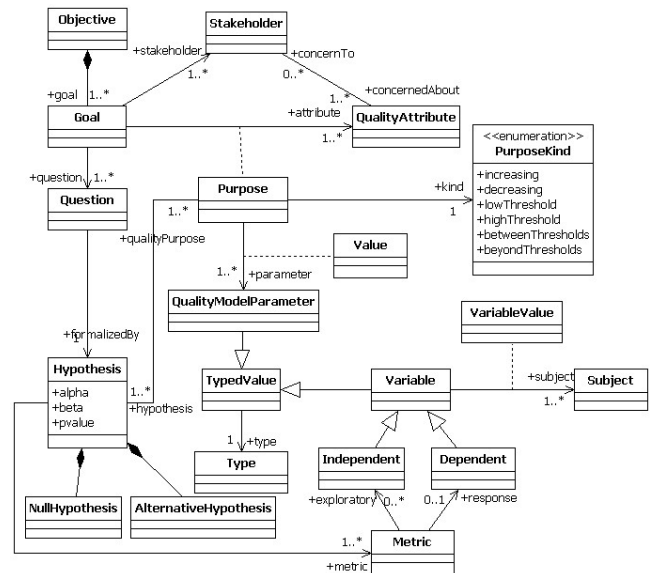


FIGURE 8 - HYPOTHESES SPECIFICATION AND VARIABLES SELECTION

B.3. Variables selection. The process of selecting appropriate variables should be guided by a goal-driven approach, such as the Goal-Question-Metric (GQM) one [29], that ties collected information to the research goals that information is intended to help achieving. This will allow preventing the collection of useless data, thus saving the corresponding resources. However, it may be useful to record data non-directly related with the main research goals, if they are expected to provide insights concerning possible side-effects of the treatments under scrutiny, which can motivate further research work.

Both dependent and independent variables selection should be based on their relevance with respect to the research goals. Each variable should be defined as clearly and unambiguously

as possible, to prevent ambiguous interpretation. This includes specifying the entity from which the measurement is taken from, the attribute being measured, the counting rule that is applied, and the unit of measurement.

B.4. Subjects selection. The target population has to be defined as clearly as possible to allow (i) devising a suitable strategy for selecting subjects and (ii) to assess the extent to which they represent the population (inference ability).

We cannot start subject selection (sampling) unless population's characteristics, including its invariants, have been clearly stated, that is, before the population is framed. Although it is desirable that collect a sample as representative as possible of the population, the selection of the sampling technique is often constrained by the available resources.

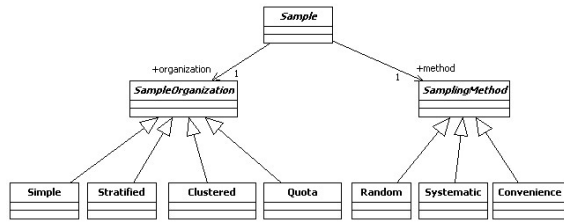


FIGURE 9 - CLASSIFICATION OF SAMPLING TECHNIQUES

With respect to the organization, sampling can be **simple** if all elements are treated equally, **stratified** if the elements are split into categories so that variations within each category are minimized, while the variations among different categories are maximized, **clustered** if the elements are grouped into clusters, or organized using **quotas** so that the elements are grouped into different categories, but then chosen in a non-random way, to ensure a pre-specified proportion among the different categories.

Regarding the adopted method, sampling can be **random** if all elements have an equal probability of being chosen, **systematic** if a selection rule is established, or based on **convenience** if elements are chosen based on their easier availability.

The combination of the simple organization with convenience sampling is the most common, in ESE.

B.5. Experiment design. The previous choices on hypotheses and variables restrict the available experiment designs. The design choice is crucial, since it conditions the valid statistical approaches that can be followed to analyze the data collected in the experiment. Three general principles that should be used when choosing an experiment design are: **randomization**, **blocking**, and **balancing** [24]. Randomization is about averaging out a factor that might otherwise influence the outcome of the test, by ensuring that observations are being made on independent random variables. When the experimenter is aware of a particular factor that may have an influence on the test but is not the factor under test, he may choose to block that effect, by creating several groups within the sample. Within each group, that factor is approximately constant, so that it has no influence on the test being performed. Balancing is about ensuring that each treatment is administered to a similar

number of subjects, to ensure a fair test. This is desirable for improving the soundness of the statistical analysis performed during the experiment.

There is no shortage of available experimental design lists, both in the context of software engineering (e.g. [24, 26, 28]), and that of other sciences (e.g. [30-32]). Simple, well-known, experiment designs are preferable as they are well documented and can be more easily replicated and understood. The usage of custom designs may require the help of a statistician, so that the design's implications are well understood [25].

An experiment design prescribes the division of the sample into a set of groups, according to some strategy. Each of those groups receives a set of interventions (either observations, or treatments). The sequencing and synchronization of such interventions, their nature, and the group definition policy, define the experimental design.

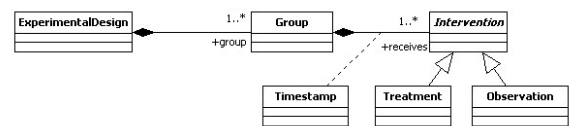


FIGURE 10 - EXPERIMENT DESIGN CONCEPTS

The **experimental design selection** starts with a decision concerning the number of groups of subjects participating in the design. With single-group designs the group assignment activity can be skipped as subjects are all assigned to the same group. There are a number of threats to internal validity associated to single-group designs, as discussed in section E.2.

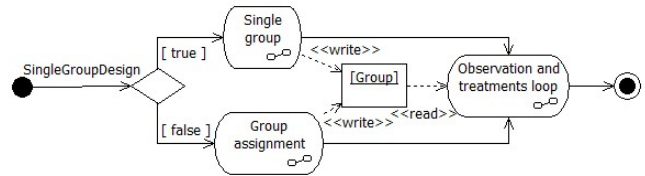


FIGURE 11 - EXPERIMENT DESIGN SELECTION

During group assignment the researcher can decide which of the group division strategies best fits the experiment goals (Figure 12).

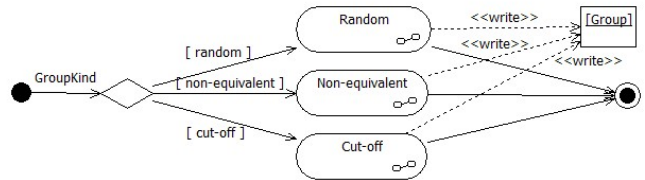


FIGURE 12 - GROUP ASSIGNMENT

Random assignment implies that every subject has an equal probability of being assigned to each of the groups. Random assignment is strong against single group internal validity threats, as well as to most multiple group internal validity threats due to the probabilistic equivalence of the groups. **Non-equivalent groups** are used very often, in quasi-experiments. A common example is when an experiment is carried out in an academic context and the groups correspond to different classrooms. This implies that the probabilistic

equivalence of the groups is lost. It is still often possible to consider the groups to be comparable, and desirable to form groups as similar as possible. Because the groups are non-equivalent, researchers must consider the additional internal validity threat of selection. Groups' difference may become a confounding effect to the analysis of the results. **Cut-off groups** are used in situations where the experimenter wishes to use a quantifiable property of the subjects as a discriminator of those subjects. The cut-off point between two groups is used as a limit between them. Subjects with a property value below the cut-off are assigned to one group, while subjects with a value above are assigned to the other. This approach is particularly useful if discontinuities are expected between the different groups. Each group is then subject to a sequence of observations and, possibly, treatments (Figure 13).

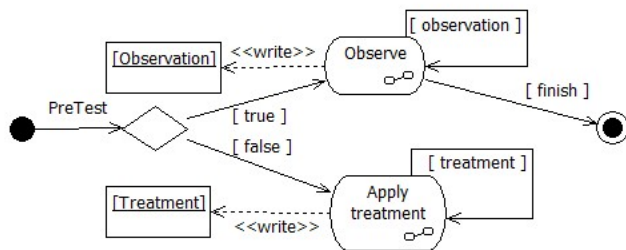


FIGURE 13 - OBSERVATIONS AND TREATMENTS LOOP

B.6. Collection process definition. The collection process definition involves defining who, when, where, and how the experimental data will be collected. Data collection activities have to be scheduled, so that potential time constraints can be considered, including the availability of participants. The instrumentation of the experiment has to be planned, to ensure the collected data quality. The driving force should be to minimize data collection effort and intrusiveness, while ensuring that data is collected in a consistent way throughout the process.

B.7. Analysis techniques. The analysis techniques chosen for the experiment depend on the adopted experiment design, the variables defined earlier, and the research hypotheses being tested. More than one technique may be assigned to each of the research hypotheses, if required, so that the results can be cross-checked later. Figure 14 presents a basic categorization of (i) data types, with respect to their scale and level of measurement, and (ii) of statistical tests techniques. The scale and level of measurement of the variables involved in the statistics tests condition the suitable tests, but a detailed discussion on this topic is beyond the scope of this paper.

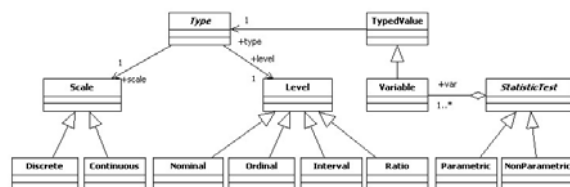


FIGURE 14 – DATA TYPES AND STATISTICAL TESTS TAXONOMY

B.8. Instrumentation. This activity involves defining the requisites, selecting, adapting or developing tools to support

measurement. The goal is to foster the comparability of the collected data by streamlining data collection consistently. This activity may also include the production of guidelines and other training material to distribute to experiment participants.

C. Experiment execution

An experiment plan can be instantiated in several different ways, due to local constraints. For instance, the number of subjects and their exact background may differ, from experiment to experiment, despite sharing a common plan. It is important to document these peculiarities, as they help grasping the “field” constraints of the experiment, in complement to those previously defined in the experiment plan. Figure 15 outlines the data collection activities.

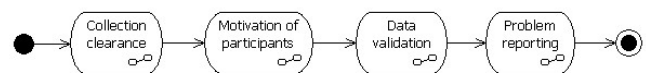


FIGURE 15 - EXPERIMENT DATA COLLECTION

C.1. Collection clearance. When the data to be used belongs to individuals or organizations and is considered sensitive, data collection clearance must be obtained before starting the collection process. Granting anonymity of participants and organizations is an approach commonly used by researchers, with the agreement of data owners.

C.2. Motivation of participants. The difficulties of recruiting professional practitioners to participate in experiments often lead to the usage of students as surrogates for those practitioners. A systematic review of controlled experiments in software engineering [1] showed that out of 5488 participants in 113 experiments, only 9.4% were professional practitioners, while 86.8% of the participants were students. The remaining participants were either faculty members, post-docs, or of a background not disclosed in the papers reporting the experiment. The report estimates that over 80% of the students were undergraduates.

In most situations, professionals participate in the experiments as part of their job (e.g. a project being used as a pilot for the introduction of a new development tool) or as part of a training course. In experiments with students, the experimental work is usually carried out within a course they follow. Their reward is often of an academic nature, such as part of the course grade, or extra credits for the student's degree. In the case of practitioners, it is important that their participation in the experiment is as non-intrusive as possible to their normal work and that the results of the experiment offer some sort of perceived added value to the organization they work for. A detailed discussion on the problems concerning the recruitment of professional participants for experiments, and possible ways to circumvent them can be found in [33].

C.3. Data collection. This is the actual enactment of the experiment. Experimenters should record information such as the schedule and effort used in the experiment by participants, so that this information can be confronted with what was previously planned. Any problems detected on the experiment

package (guidelines) should also be registered, so that it can be improved in further replications of the experiment. Special events concerning the experiment, such as subject's mortality (subjects that are removed from the experiment for some reason) must be recorded for further analysis. The potential impact of mortality on the experiments results should also be assessed. Patterns on the mortality of subjects may help uncovering factors which are important to the experiment but are not addressed by the adopted design. These factors should be considered, when analyzing the threats to the validity of the experiment.

C.4. Data validation. This process aims at ensuring that the experiment data has been collected correctly. Problems with data collection can result from, for instance, erroneous performance of collection tools, misinterpretation of data collection forms by the experiment participants, or deviations from the planned experimental protocol. Data quality is essential for inference purposes. This validation activity may involve not only the researchers conducting the experiment, but also the participants. The latter can help clarifying data that is found likely to be incomplete, or incorrect.

C.5. Problem reporting. All deviations from the original plan should be recorded. Some details may have been insufficiently covered or even not considered at all in the planning phase. Identifying those problems, and how they were dealt, facilitates experiments replication, and is a relevant step toward identifying potential validity threats in the results.

D. Data analysis

This activity follows data collection, since its target is the collected data (Figure 16). It involves the description of the data set, its reduction, and testing the hypotheses defined during the experiment plan.

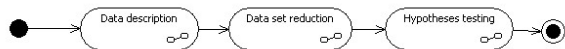


FIGURE 16 - EXPERIMENT DATA ANALYSIS

D.1. Data description. The data collected in the experiment should be analyzed by using, in a first moment, adequate descriptive statistics. The set of adequate statistics varies with respect to the target variables under scrutiny. For continuous variables, the count of observations, mean value, median, mode, minimum, maximum, and standard deviation are often collected. For discrete variables, a frequency analysis is adequate. Data description helps in understanding its central tendency and dispersion. If the sample is split into several groups, to accommodate the chosen experimental design, the data description should be performed for each of those groups individually.

D.2. Data set reduction. Data description allows detecting atypical cases, such as incorrect, outlier or extreme values. Their cause should be identified, since this may allow improving the collection process. To avoid biasing the subsequent analysis performed on the data set, the removal of

atypical cases should be considered before performing hypotheses testing.

D.3. Hypothesis testing. This activity consists in performing the statistical tests that will assess the hypotheses defined in the experiment plan. This involves checking that the pre-conditions for the tests that will be performed are met. Sometimes, restrictions imposed by the data obtained during the experiment may induce adjustments in the hypotheses being tested. If so, such changes should be clearly documented. Interesting discussions on hypotheses evaluation in the context of software engineering can be found in [23-25]. They all convey the notion that the soundness of the hypothesis test has to be as verifiable as possible by external observers reading the experiment's report. This is only possible if a detailed description of the tests results, their probability, degrees of freedom, direction, and test power are reported. This level of detail is essential in order to perform meaningful comparisons of results from tests applied in independent experiments.

E. Results packaging

Once the experimental work per se is finished, it is essential to package the results so that they can be used either within the context of the organization that sponsors the experiment, or by the community. This involves documenting the whole experimental process, as discussed in the previous sections, and including a discussion on the results achieved with the experiment. The discussion should focus on aspects such as the interpretation of the results, the limitations of the study, the inference that can be made, and the identification of the learned lessons.

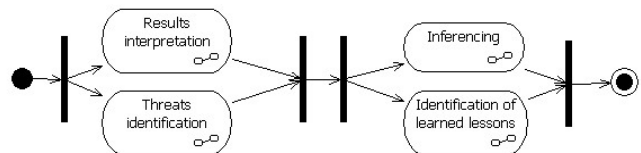


FIGURE 17 – EXPERIMENT RESULTS PACKAGING

E.1. Results interpretation. This activity concerns the analysis of the outcome of the tests, anchored on the theory that is being assessed through them. When the tests do not confirm the theoretical assumptions, identifying the causes that lead to that failure can be used as a stepping stone to allow the refinement of the theory, or even the construction of a new theoretical framework to explain the object or process under test.

E.2. Threats identification. Decisions concerning activities such as sampling, or experimental design selection have an impact on the validity of the results obtained in the experimental process. While packaging the experimental results, one should look back to the whole process and clearly identify the potential threats to the validity of those results. Furthermore, one should discuss the measures that were in place to address each of those threats.

The objective of threats identification is to document potential weak spots of the experimental work being

described. Rather than a depreciation of the value of the experimental work, this analysis can be viewed as an active and systematic approach to identifying opportunities for further complementary studies that, as a family of related studies, can contribute to the software engineering body of knowledge. See [24] for a detailed discussion on common threats to validity in ESE.

E.3. Inferencing. Considering all the identified threats to validity, the experimenters have to estimate how the results obtained in the experiment are expected to hold beyond the experiment's sample (i.e. by the population).

E.4. Identification of learned lessons. During the whole experimental process, the practical details of conducting that process, including potential gaps or impractical design decisions in the experimental protocol should be registered, along with the approach followed to circumvent these problems. This information is valuable in experiment replications, as a mitigation of the tacit knowledge problem.

IV. COMPARISON WITH RELATED WORK

A. Experiment conduction guidelines

The process and data model presented in this paper supports the concepts conveyed in empirical research guidelines, such as [24, 25]. While more detailed discussions on each of the process steps can be found in those documents, here we favored the integration of all these concepts into a single model.

B. Experiment reporting guidelines

Our process description captures the main activities required in *Jedlitschka and Pfahl's* reporting guidelines for controlled experiments in software engineering [27, 34]. While the deliverables of each activity can be mapped to their proposal, we adapted our model so that other (weaker, but often more feasible) types of empirical work, such as observational studies and quasi-experiments, can also be supported by a process instantiating our model.

C. Models for representing experimental data

Concerning quantitative data representation, *Kitchenham and Hughes* proposed a method for specifying models of software data sets that capture the definitions and relationships among software measures [35]. The rationale is that without proper safeguards, it is very difficult to ensure that data from different sources can be safely combined and analyzed. Software measurements have to be fully defined, rather than just named, to allow for the repeatability of their collection. *Kitchenham and Hughes's* work, is focused on the storage of the information concerning the metrics definitions and values (including collected values, estimates and targets), at different levels of granularity.

Garcia et al. defined a metamodel for software measurement that represents concepts including the measurement activities, measures definitions, and metrics definitions [36]. This metamodel was an attempt to develop a unified terminology for describing software measurement

concepts, involving researchers from Spain and Argentina, although the authors admit that no complete consensus was achieved, due to different research focuses.

Jeusfeld *et al.* proposed a metamodel for modeling quality management issues in the context of data warehouses [37]. Their metamodel is inspired by the GQM approach [29], and covers goal specification, the construction of queries for assessing the goals, and the definition of metrics to support those queries.

Our work shares the concern of having a metamodel description upon which we can describe experimental work, but covers a wider spectrum of experimental concepts and adds the elements of dynamics description (in this paper) and of formal metrics definition and collection (see, for instance, [38-42]).

V. CONCLUSIONS

This paper proposed a model for the experimental software engineering process, using UML2 Activity Diagrams, that supports the definition and enactment of the activities, and corresponding deliverables, involved in ESE practice. Its major novelty is the integration, in a homogeneous framework, of the contributions of other researchers along three threads: (i) experiment conduction guidelines, (ii) experiment reporting guidelines and (iii) models for representing experimental data.

The proposed process model captures the best practices in the area and is aligned with recent proposals for experimental data dissemination. It can be used as a guideline for practitioners involved in leveraging data collection activities to improve the software process in their organizations, both in industrial and academic contexts. It may also be used as a framework for supporting experiments comparison, which was identified as one major need for future software engineering research. The full model is presented in the appendix.

ACKNOWLEDGMENT

This research work was developed in the realm of the STACOS (STandard-based COoperative Software) project (POSI/CHS/48875), financed by the Portuguese Foundation for Science and Technology.

REFERENCES

- [1] D. I. K. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Transactions on Software Engineering*, vol. 31, pp. 733-753, 2005.
- [2] R. L. Glass, I. Vessey, and V. Ramesh, "Research in software engineering: an analysis of the literature," *Information and Software Technology*, vol. 44, pp. 491-506, 2002.
- [3] V. Ramesh, R. L. Glass, and I. Vessey, "Research in computer science: an empirical study," *Information and Software Technology*, vol. 70, pp. 165-176, 2004.
- [4] W. Tichy, P. Lukowicz, L. Prechelt, and E. A. Heinz, "Experimental Evaluation in Computer Science: A Quantitative Study," *Journal for Systems and Software*, vol. 28, pp. 9-18, 1995.
- [5] M. V. Zelkowitz and D. Wallace, "Experimental Validation in Software Engineering," *Journal of Information and Software Technology*, vol. 39, pp. 735-743, 1997.

- [6] W. Tichy, "Should Computer Scientists Experiment More?," *IEEE Computer*, pp. 32-40, 1998.
- [7] B. A. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based Software Engineering," presented at 26th International Conference on Software Engineering, Edinburgh, Scotland, 2004.
- [8] T. Dybå, B. A. Kitchenham, and M. Jørgensen, "Evidence-based software engineering for practitioners," *IEEE Software*, vol. 22, pp. 58-65, 2005.
- [9] M. E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, vol. 15, pp. 182-211, 1976.
- [10] M. E. Fagan, "Advances in Software Inspections," *IEEE Transactions on Software Engineering*, vol. 12, pp. 744-753, 1986.
- [11] A. Porter and P. Johnson, "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies," *IEEE Transactions on Software Engineering*, vol. 23, pp. 129-145, 1997.
- [12] F. B. Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality," presented at 3rd International Software Metrics Symposium (Metrics'96), Berlin, Alemanha, 1996.
- [13] D. Budgen and M. Thomson, "CASE tool evaluation: experiences from an empirical study," *Journal of Systems and Software*, vol. 67, pp. 55-75, 2003.
- [14] W. E. Deming, *Out of the Crisis*, 1st ed. Cambridge, MA, EUA: The MIT Press, 2000.
- [15] V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology," presented at 1st Pan Pacific Computer Conference, Melbourne, Australia, 1985.
- [16] M. B. Chrissis, B. Broekman, S. Shrum, and M. Konrad, *CMMI: Guidelines for Process Integration and Product Improvement*: Addison-Wesley Professional, 2003.
- [17] B. Henderson-Sellers, F. Stallinger, and B. Lefever, "The OOSPICE Methodology Component: Creating a CBD Process Standard," in *Business Component-Based Software Engineering, The Kluwer International Series in Engineering and Computer Science*, F. Barbier, Ed. Boston Hardbound: Kluwer Academic Publishers, 2002.
- [18] A. Brooks, "Meta Analysis—A Silver Bullet—for Meta-Analysts," *Empirical Software Engineering*, vol. 2, pp. 333-338, 1997.
- [19] J. Miller, "Applying Meta-Analytical Procedures to Software Engineering Experiments," *Journal of Systems and Software*, vol. 54, pp. 29-39, 2000.
- [20] R. Conradi and A. I. Wang, "Empirical Methods and Studies in Software Engineering: Experiences from ESERNET," in *Lecture Notes in Computer Science*, vol. 2765, G. Goos, J. Hartmanis, and J. v. Leeuwen, Eds. Heidelberg: Springer Berlin, 2003.
- [21] A. Jedlitschka and M. Ciolkowski, "Towards Evidence in Software Engineering," presented at International Symposium on Empirical Software Engineering (ISESE'04), 2004.
- [22] F. Shull, V. R. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri, "Replicating software engineering experiments: addressing the tacit knowledge problem," presented at 2002 International Symposium on Empirical Software Engineering (ISESE'02), 2002.
- [23] J. Singer, "Using the American Psychological Association (APA) Style Guidelines to Report Experimental Results," presented at Workshop on Empirical Studies in Software Engineering, Oxford, England, 1999.
- [24] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, vol. 6. Boston, EUA: Kluwer Academic Publishers, 1999.
- [25] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 28, pp. 721-734, 2002.
- [26] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*: Kluwer Academic Publisher, 2001.
- [27] A. Jedlitschka and D. Pfahl, "Reporting Guidelines for Controlled Experiments in Software Engineering," presented at 4th International Symposium on Empirical Software Engineering (ISESE 2005), Noosa Heads, Australia, 2005.
- [28] V. R. Basili, "The role of experimentation in software engineering: past, current, and future," presented at 18th International Conference on Software Engineering, Berlin, Germany, 1996.
- [29] V. R. Basili, G. Caldiera, and D. H. Rombach, "Goal Question Metric Paradigm," in *Encyclopedia of Software Engineering*, vol. 1, J. J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 469-476.
- [30] T. D. Cook and D. T. Campbell, "The design and conduct of quasi-experiments and true experiments in field settings," in *Handbook of industrial and organizational psychology*, M. D. Dunette, Ed. Chicago: Rand MacNally, 1976, pp. 223-326.
- [31] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 2nd ed: SAGE Publications, 2003.
- [32] W. M. Trochim, "The Research Methods Knowledge Base, 2nd Edition. WWW page (<http://trochim.human.cornell.edu/kb/index.htm>), 2006.
- [33] H. C. Benestad, E. Arisholm, and D. I. K. Sjøberg, "How to Recruit Professionals as Subjects in Software Engineering Experiments," presented at Information Systems Research in Scandinavia (IRIS), Kristiansand, Norway, 2005.
- [34] A. Jedlitschka and M. Ciolkowski, "Guidelines for Empirical Work in Software Engineering," Fraunhofer Institute for Experimental Software Engineering, Technical Report IESE-Report No. 053.05/E (Version 1.0), August 2005.
- [35] B. A. Kitchenham and R. T. Hughes, "Modeling Software Measurement Data," *IEEE Transactions on Software Engineering*, vol. 27, pp. 788--804, 2001.
- [36] F. Garcia, F. Ruiz, M. Bertoa, C. Calero, M. Genero, L. Olsina, M. Martín, C. Quer, N. Tondori, S. Abrahao, A. Vallecillo, and M. Piattini, "Una Ontología de la Médiación del Software," Universidad de Castilla-La Mancha, Technical report UCLM DIAB-04-02-2, February 2004.
- [37] M. A. Jeusfeld, C. Quix, and M. Jarke, "Design and Analysis of Quality Information for Data Warehouses," presented at International Conference on Conceptual Modeling / the Entity Relationship Approach, 1998.
- [38] F. B. Abreu, "Using OCL to formalize object oriented metrics definitions," INESC, Software Engineering Group, Technical Report ES007/2001, May 2001.
- [39] A. L. Baroni and F. B. Abreu, "Formalizing Object-Oriented Design Metrics upon the UML Meta-Model," presented at Brazilian Symposium on Software Engineering, Gramado - RS, Brazil, 2002.
- [40] A. L. Baroni, C. Calero, M. Piattini, and F. B. Abreu, "A Formal Definition for Object-Relational Database Metrics," presented at 7th International Conference on Enterprise Information System, Miami, USA, 2005.
- [41] M. Goulão and F. B. Abreu, "Validação Cruzada de Métricas para Componentes," *IEEE Transactions Latin America*, vol. 3, 2005.
- [42] M. Goulão and F. B. Abreu, "Composition Assessment Metrics for CBSE," presented at 31st Euromicro Conference - Component-Based Software Engineering Track, Porto, Portugal, 2005.

APPENDIX – EXPERIMENTAL PROCESS OVERVIEW

