An Environment to Support Large Scale Experimentation in Software Engineering

Guilherme H. Travassos, Paulo Sérgio Medeiros dos Santos, Paula Gomes Mian
Arilo Cláudio Dias Neto, Jorge Biolchini
Systems Engineering and Computer Science Program
COPPE/UFRJ - Brazil
{ght, pasemes, pgmian, acdn, jorgebio}@cos.ufrj.br

Abstract

Experimental studies have been used as a mechanism to acquire knowledge through a scientific approach based on measurement of phenomena in different areas. However it is hard to run such studies when they require models (simulation), produce large amount of information, and explore science in large scale. In this case, a computerized infrastructure is necessary and constitutes a complex system to be built. In this paper we discuss an experimentation environment that has being built to support large scale experimentation and scientific knowledge management in Software Engineering.

1. Introduction

Software Engineering represents a broad field. Its importance has grown continuously due to increase of humankind's dependency on software systems. One of its aims concerns with how software technologies (processes, methods, techniques and tools) can be used to improve development processes, product quality and team productivity while building software. However, even with all the development and knowledge acquired related to the Software Engineering field, it is still common to decide about the use or development of software technologies just based on personal opinion or commercial interest.

Decisions related to software technologies usage should not be directed or based neither on intuition neither speculation. In fact, the decisions should as much as possible rely on concrete evidences, results of experimental studies, represented by observations of_or experimentation with the real world and its measurable behaviors [8].

This scientific approach has been used in different knowledge fields, such as Physics, Classical Engineering, Biology, and Medicine for several decades. Some other areas, such as Nuclear Physics, Biomedical Informatics and Earth Sciences have demonstrated to be dependent on an integrated computerized infrastructure distributed on the Web to apply their scientific approaches and manage their experimentation processes [40]. In more recent areas, such as Computational Biology, to accomplish experimentation a computerized infrastructure is crucial. Some experiments can be represented by studies involving multiple combinations of data and software programs, with complex management whether manually accomplished by the researcher. In this case, the expected subject behavior is directly related to his/her ability to produce all feasible combination of data and procedure calls to run the study. Therefore, the use of an experimentation environment is clearly justified to reduce clerical activities and concentrate the human interaction in the study starting and ending points. At the beginning, the researcher configures the scientific workflow that will guide the experimentation process. At the end, the researcher will be dealing with new available data for analysis [11].

All of these types of complex experimentation scenarios reinforce the idea that experimentation can get benefits whether using appropriate computerized resources. In fact, when the researcher needs to explore computer models (simulation or visualization) or even accomplish the study in the large, involving more researchers eventually geographically distributed, computerized infrastructures represent one of the requirements [48]. As one can see, experimentation environments represent complex systems that need to be characterized and constructed. This should be also considered when dealing with the software engineering field and the experimentation activities related to the construction and evaluation of software systems and software technologies [8,26].

Accordingly to Basili [7], Software Engineering is big science. Experimentation has been used as a mechanism to acquire knowledge through a scientific approach based on measurement of phenomena in different Software Engineering areas. In fact, experimentation can provide some indication about

how such software technologies can contribute for quality improvement. Thus, the results of the studies executed in different development scenarios can be used to compose a criteria set to support the decision making process regarding software technologies usage [8].

However, not different from other related scientific fields, the accomplishment of experimental studies in Software Engineering is time consuming, hard task, and produces great volume of information and knowledge with complex management [45]. Hence, some studies strongly depend on a computerized infrastructure to support its processes. For these studies, the usage of web resources to support the accomplishment of experimental studies can be an option to help in obtaining more realistic experimentation contexts, closed to the working industry configuration and allowing to deal with some logistical issues, such as the accomplishment of tasks by geographically distributed teams. It could also benefit the recruitment, selection and participation of geographically distributed subjects. Arisholm et al. [2] define such experimentation arrangements as large scale experiments, which can lead to the improvement of population sampling power and the increase of reliability on the observed behaviors.

This prospective scenario of research perspectives regarding experimentation in the large in Software Engineering has motivated the Experimental Software Engineering group at COPPE/UFRJ (http://www.cos.ufrj.br/~ese) to work on the definition, designing and building of an experimental Software Engineering Environment (eSEE). It is represented by a computerized infrastructure to support large-scale experimentation in Software Engineering. eSEE provides a set of facilities to allow geographically distributed software engineers and researchers to accomplish and manage experimentation processes as well as scientific knowledge concerned with different study types through the web.

An experimentation environment, such as eSEE, represents a complex system. Its development requires special attention to topics such as the experimentation domain, software architecture and the engineering of the whole system, security, and the way that computerized facilities and tools must be integrated into the environment, making all of them available to the researchers. Besides Software Engineering, eSEE facilities have been used to support experimentation in other science fields. For instance, it has been used to manage the scientific workflow of an integrated workbench for analysis, modeling, simulation and visualization of sedimentary basins and petroleum systems in deep waters [19].

The objective of this paper is to describe the main features of eSEE, an experimentation environment to support large scale experimentation and scientific knowledge management in Software Engineering. It is composed by seven sections. The first one comprises this introduction. Section 2 discusses experimentation in Software Engineering, highlighting some of the study's related main concepts. In the sequence, section 3, primary and secondary studies are explained. Section 4 presents eSEE and its facilities. In section 5 the way knowledge is considered in eSEE is described. Next, a complementary way to explore experimentation through eSEE is introduced. Finally, Section 7 summarizes and concludes the paper.

2. Experimentation in Software Engineering

The importance of experimentation in Software Engineering was firstly described by Basili et al. [4] in the 80's. Since then, an increasing interest in experimentation on the field can be observed [49, 54]. An explanation for this can be due to the viewpoint that Software Engineering must have strong foundations as a scientific engineering discipline, and that the techniques to improve software development and to evolve processes must be available to professionals. For instance, there is an increasing agreement in the Software Engineering community that experimentation is necessary to develop or to improve software development and maintenance processes, methods and tools [45]. The classical method for identifying cause-effect relationships is to conduct controlled experiments where only few variables can vary. However, differently from other scientific fields, experimental research in Software Engineering must deal with software processes and products, taking into account a large number of problems, context variables (sometimes unknown) and characteristics, being a complex activity [5].

The development of methods and technologies to support experimentation in Software Engineering represents an ongoing research field with several open questions. Among the main efforts we can find the definition itself of an experimentation process. In this sense, there is still no consensus and some additional research efforts need to be invested [25].

Differently from other production areas, software developers neither produce the same products all the time nor take part at the same development team in all software projects. Each developed software effort is

different from the previous one [5]. In complement, human factors impact the engineering of software; there are several other context variables that make critical the execution of experimental studies despites being an academic or industrial environment.

Besides, from the experimental viewpoint, for instance, it is important to highlight how representative the study population could be. Usually the number of subjects is not enough to configure a real population sample. Some experimental studies do not have a number of subjects that can guarantee a software developer population to be a feasible approximation for statistical analysis. There is also the difficulty regarding random sampling. Usually, researchers need to get the subjects by convenience, choosing those ones that are taking part in a course, software project, training and so on. Controlled experiments in Software Engineering often involve students solving small pen and paper tasks in a classroom setting. A major criticism regarding such studies is the lack of realism, which may defer technology transfer from the research community to the industry. The studies would be more realistic if run on real tasks, on real systems, with target population software professionals that are representative of the technology as target population, and using their usual development technology in their usual working environment [8].

From the organization viewpoint, it is not easy to combine the execution of an experimental study with the academia/industry objectives and daily tasks. For instance, it is risky for an academic or industrial project to allocate developers, time or equipments to take part in studies whether they are out of project focus. In the academic context, there is a need to organize the study context, objectives and schedule with the content of a course, allowing the participation of the students as subjects. The lack of information and management of these factors can make hard the execution of the study and can introduce threats for both its repeatability and validity of results [53].

Despite all these issues, which represent research challenges, experimentation is a reality in Software Engineering. It has supported the establishment of an experimental software engineering research community and the evolution of the Software Engineering field [6]. Besides, it has motivated other Computer Science areas to also explore the scientific approach [23]. However, in a similar way to researchers from other knowledge fields, Software Engineering researchers and practitioners need to efficiently integrate and validate data, generated by experimentation, to provide a solid basis for rational decision-making and production of evidences.

The importance of applying an evidence-based methodology in the scientific research can be illustrated by the experience that has been gathered in Medicine. For many years, the medical area had been full of scientific reviews accomplished without formal methods to identify, evaluate, and summarize the existing information in the scientific literature. A potential consequence of applying such informal methodology could be observed in the late 80's. At that time, studies accomplished to evaluate the quality of some medical publications called the attention to the observed poor scientific quality [14]. Some studies found out that, on one hand, the failure in organizing medical research through systematic reviews could be affecting human lives [15]; on the other hand, clinical judgments by some specialists were considered inadequate when compared against experimental results from systematic reviews [1]. Since then, the recognition of the necessity for conducting literature reviews in a systematic way has been grown fast. This fact can be proved by the great quantity of formal reviews published each year in the medical area [41]. Such type of reviews, also known as systematic reviews, are rigorous methodological processes looking for relevant indications that can result in the identification of evidences in the research field under investigation.

Kitchenham et al. [30], as far as we know, were the first researchers to draw a parallel between Medicine and Software Engineering regarding an evidence-based approach. According to these researchers, Evidence-based Software Engineering must provide means over which best evidences from research can be integrated with practical experience and human values in the process of decision making regarding both software development and maintenance. We have observed that to achieve an adequate level of evidence regarding a specific technology under investigation, the Evidence-based Software Engineering may basically adopt two types of studies: primary and secondary ones [33].

3. Primary and Secondary Studies in Software Engineering

The primary studies conduction is important to contribute to a larger body of scientific knowledge, by refining techniques, methods, and tools, and generating new hypotheses. For primary studies, one means the conduction of studies with the goal of characterizing the use of a specific technology in a specific context. In this category, one can find the experimental studies, as experiments, case studies, and surveys [53].

Accordingly to Travassos and Barros [52] these experimental studies can be classified in four categories:

• *In vivo*: such experimental studies involve people in their own environments. In Software Engineering, *in vivo* studies are executed in software development organizations throughout the development process and under real conditions;

• *In vitro*: such experimental studies are executed in controlled environments, such as controlled communities or laboratories. Most *in vitro* studies are executed in universities, research centers or among selected groups of software development organizations;

• *In virtuo*: these studies involve the interaction among participants and a computerized reality model. In these experimental studies, the environment behavior with which subjects interact is described as a model by a computer program. In Software Engineering, these studies are usually executed in universities and research laboratories characterized by small groups of subjects manipulating simulators;

• *In silico*: these experimental studies are characterized by having both the subjects and the real world being described by computer models. In this case, the environment is fully composed by numeric models into which no human interaction is allowed. Because i*n silico* studies need a large amount of SE knowledge they are still rare in Software Engineering. For instance, we can find *in silico* studies applied to software usability experimentation, such as software performance characterization or software process simulation.

Primary studies can be used to observe behaviors, reveal trends in the field, testing hypotheses and evaluate software technologies. For instance, the methodology introduced by Shull et al. [44] covers the conduction of primary studies to evaluate software technologies.

Secondary studies can be defined as a means of identifying, evaluating, and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. The systematic review is a type of secondary study [14, 31]. In this sense, results obtained by several correlated primary studies act as a source of information to be investigated by secondary ones (Figure 1).

Furthermore, secondary studies cannot be considered as an alternative approach to the primary production of evidence, which is represented by the primary studies [9]; secondary studies are in fact complementary to primary studies in this process. The precision and reliability resulting from secondary studies accomplishment contributes both to improve and direct new research topics to be investigated by primary studies, in an iterative way.

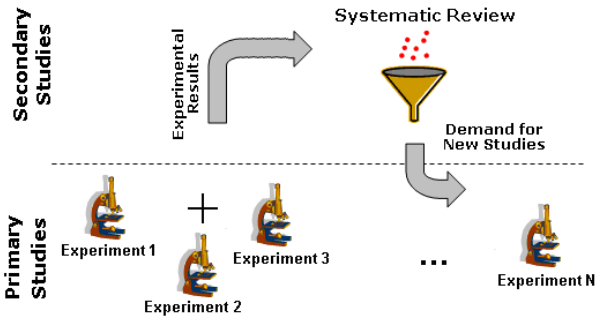

Figure 1. Interaction between studies [34]

Conducting Systematic Reviews in Software Engineering (SE) represents a major methodological tool to scientifically improve the validity of the assertions that can be made in the field and, as a consequence, the reliability degree of the methods that are employed for developing software technologies.

A systematic review is more complete, fair, with a greater scientific value when compared to *ad-hoc* literature reviews, representing a method to identify, evaluate and interpret the pertinent research about a particular research question. Other specific reasons justifying the use of systematic review in software engineering are concerned with summarizing evidences about a specific theory or technology; identifying research area gaps, highlighting those ones needing further investigation, and; providing grounds for new research activities and projects.

Usually, a protocol must be developed by specifying the steps and criteria that shall be used to undertake the review, which is started by formulating its research question. Pai et al [42] identify that a well-formulated research question can be built by exploring four perspectives, identified as **P**opulation, **I**ntervention, **C**omparison, and **O**utcome:

P: The sample population (e.g. the disease group, or a spectrum of the healthy population);

I: The study factor (e.g. the intervention, diagnostic test, or exposure);

C: The comparison intervention, if applicable;

O: The outcome.

Our first initiatives of conducting systematic reviews used the work of Kitchenham [31] and the protocol example by Mendes and Kitchenham [35] as the reference material. The students' motivations to conduct systematic reviews were diverse at that moment. Some of them intended to collect indicators of

the existence or use of some technique. Others aimed at identifying gaps in the SE research area, suggesting new works in the field, or pointing the current context of a research topic. The work by Conte et al. [16] regarding Web-based Systems Development Processes represents a SR aimed at characterizing processes that are used to develop web applications. The objective was to identify the current state of the art of that particular research topic.

Based on our acquired experience with the initial systematic reviews [38], we have described a process and a template for systematic reviews [9, 39] in order to guide researchers in undertaking systematic reviews in the Software Engineering domain. Some of them resulted in technical papers published elsewhere [3,21,32,47]. From our experience in undertaking Systematic Reviews we have observed that, for characterization reviews, usually one of the PICO perspectives can not be considered. For instance, most of the characterization reviews (initial reviews) do not perform any kind of comparison. The EBSE project [22] identifies this sort of review as being a systematic mapping study. However, we prefer to identify it as a *quasi*-systematic review. This conceptual and terminological choice is derived from the fact that such review must explore the same rigor and formalism for the methodological phases of protocol preparation and running, but no meta-analysis in principle can be applied. Even so, they are important to establish an initial knowledge baseline for future primary study comparison and can produce valuable knowledge.

4. An environment to support experimentation in Software Engineering

The idea concerned with this experimentation environment for Software Engineering was initially introduced at the ISERN - International Software Engineering Research Network annual meeting in 2003 [51] and lately evolved additional research works its requirements [37], architecture [20], experimentation process [12] and environment configuration facilities [43].

eSEE – ***e****xperimental* ***S****oftware* ***E****ngineering* ***E****nvironment* – is a computerized infrastructure able to create environments to support scientific knowledge management throughout the software engineering experimentation process, including the definition, planning, execution and packaging of primary and secondary studies in Software Engineering [37]. The eSEE aims at supporting researchers in accomplishing primary and secondary studies through the web. For realizing this, a set of basic requirements must be fulfilled [12]:

- Having integrated experimentation support tools, which shall perform similarly as a Software Development Environment (SDE);
- Being composed as a Web System, allowing its use in different localities and by inter-institutional research teams;
- Making extensively use of e-services and,
- Providing knowledge management facilities, since Experimental Software Engineering is a knowledge intensive area.

In the next subsections the main features of eSEE are described.

4.1 eSEE conceptual model

The eSEE's conceptual model has been basically organized in three abstraction levels: meta, configured and execution. These three abstractions levels relate to the primary and secondary study knowledge and correspondent environment facilities that must be used and made available for the researchers:

- Meta-level: knowledge for any kind of study. The meta-level contains common knowledge regarding experimental software engineering and its studies, including Software Engineering knowledge. At this level, a standard experimentation process (SEP) can be defined. The SEP represents the basis to create instances of the standard process for each type of study;
- Configured-level: knowledge for each type of experimental study. At this level, the creation of specific environments can be accomplished both by choosing the study type and its correspondent standard experimentation process and by adding specific study's characteristics. An environment instance, then, can be generated with the specific knowledge to that study type;
- Execution-level: knowledge for a specific study. It supports the definition, planning and execution of a specific primary or secondary study type. For each new study, one Execution Environment must be created to manage the experimentation process and its correspondent scientific knowledge.

4.2 eSEE architecture

The conceptual model inspired the design of the eSEE architecture [20], which is composed by three distributed macro-components. This architecture aims at suggesting a solution for integrating the different conceptual layers concerned with the experimental

software engineering processes modeling, instantiation and running. These distributed macro-components are represented in eSEE by the following software modules: Meta-Configurator (MC), Instantiation Environment (IE) and Execution Environment (EE) (Figure 2).

The Meta-Configurator, as illustrated in Figure 2, deals with the meta-description activities including the definition of process and document models and configuration of e-services [29], which will respectively represent the general scientific workflow, the instruments and the services that will be used throughout the experimentation process. The process and document models are defined by following meta-models (represented by XML schemas), which aggregate knowledge from different primary and secondary studies. Each element is stored in a repository using a XML format for further linkage among them. All eSEE repositories use native XML storage technology - eXist Database (http://exist.sourceforge.net/) - to facilitate the XML data query and manipulation.

The first component, Process Modeling Component [46] (available at http://ese.cos.ufrj.br/MetaConfigurador/), allows the reuse of other process models and knowledge regarding previous studies. An experienced software engineer can formalize knowledge regarding the experimentation process through the definition of questions, which involve decisions of including or excluding tools, roles and activities regarding the study. These questions will be used to guide the researcher in the configuration of the experimentation process for its instantiation. The second one, Document Modeling Component, is used to elaborate the instruments and documents produced/consumed throughout the Experimentation Process. Finally, the last component, E-Service Configuration Component, is used when the service needs some adaptation in its interface to be used by the environment. Thus, this component allows the definition of a new service interface to make it available to be used into the eSEE. To achieve this, the component creates a new service that works like an adapter between the infrastructure and the original service. The Document Modeling Component and E-service Configuration Component are not currently browser based.

The Instantiation Environment macro-component provides support for the definition and instance creation of an Execution Environment, based on the experimentation process, documents and services previously defined by using the Meta-Configurator. At this level, there is a Process Configuration Component [46] (available at http://ese.cos.ufrj.br/Configurador/) to analyze an initial process model, adapting it for a specific study. The Associations Mapping Component (available at http://ese.cos.ufrj.br/Mapeador/) aims at establishing the relationships among the experimentation process's activities, their correspondingly produced/consumed documents and the respective configured services that should be used to accomplish an experimentation task. This set of relationships is called an Association Map. It is stored in a specific repository, from where the Instantiation Component will be able to use the information to create an instance of a specific Execution Environment for the study.

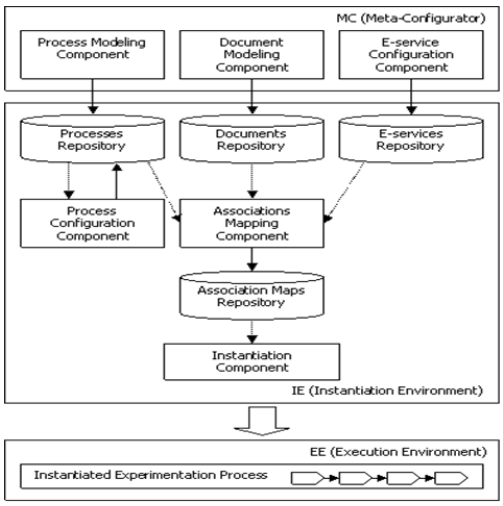

Figure 2 - Basic Concepts of eSEE architecture

Finally, the Execution Environment macro-component is responsible for the enactment of the experimentation process. It allows the monitoring, controlling, and execution of the experimentation process activities based on the conditions and restrictions imposed during its configuration [43].

The current implemented eSEE´s facilities allow the instantiation of experimental environments to support *in vitro* studies for geographically distributed research teams. Also available is a set of facilities to deal with systematic reviews. Some additional research efforts have been invested to make eSEE able to deal with *in vivo*, *in virtuo* and *in silico* experimental studies. To evaluate the feasibility of eSEE's infrastructure, a prototype has been built [19]. In this first prototype, simple documents and processes models were produced without tools to support it, and the service configuration activities were not considered due to the lack of e-services to use in this context. Using this experience, Santos and Travassos [43] improved the prototype towards both the implementation of the

Associations Mapping Component and the implementation of the Process Configuration Component, allowing the configuration of the process model to be based on answers given by the engineer that is responsible for the study. This prototype was used to support the accomplishment of an experimental study regarding a software inspection technique with positive results accordingly the participants [13, 34].

Currently, we are concentrating efforts on constructing the component that will be responsible for the eSEE's integration with the configured services allocated during the mapping task. This component, called Services Manager, is an EE subcomponent. By using an association map the Services Manager is able to define which configured services have to be invoked to support an activity that is being executed in the Execution Environment.

The Services Manager Component uses data integration to communicate with configured services. This type of integration requires that these services follow the eSEE data model. Doing so, most part of the services need the construction of a specific wrapper component that performs data translation between eSEE and the configured service. A semantic data treatment, using ontologies, for instance, could facilitate this communication since the same information is often represented in different ways among different services. Furthermore, we are considering in the future evolving the control/process integration in order to be able to combine configured services. This would allow eSEE to provide new functionalities from existing services. Some initial investigation indicates that this integration type will be important mainly in studies involving simulation [11].

5. Dealing with Knowledge in eSEE

The eSEE must manage the software engineering experimentation process, including knowledge acquired when defining, planning, executing and packaging studies. eSEE makes available experimentation process models, experiment packages, data representation standards, knowledge management facilities, tools, services, quality, and computerized models (simulators) for different study's types [37].

In order to organize knowledge to perform studies it is necessary to formalize the common terminology of the involved concepts, represented by an ontology [10]. Ontologies can be used to enable multiple target applications or humans to have access to heterogeneous sources of information that are expressed by using diverse vocabularies or inaccessible formats. Ontology of a given domain can provide a vocabulary for specifying requirements for one or more target applications. In fact, ontology is used as a basis for software specification and development, allowing knowledge reuse. Also, ontologies are applied to search an information repository looking for desired resources, improving precision and reducing the overall amount of time spent in the searching [24].

An environment such as eSEE would also benefit from the definition of ontology to organize knowledge about Experimental Software Engineering. This ontology can be core to knowledge retrieval, supporting the identification of the general study types characteristics and the common knowledge regarding software engineering studies, as well as to allow the communication amongst users and tools, opening the opportunity to explore science in large scale concepts into the experimentation domain.

Ontologies are used in eSEE, for instance, to represent the structure of experimental processes in the Meta-Configurator component through software process ontology [39]. In addition, the work of Biolchini et al. [10] proposed a scientific research ontology to support systematic review knowledge organization that is being generated from the conduction of secondary studies in eSEE. It is being integrated to the eSEE conceptual infrastructure and represents an initial step towards a wider Experimental Software Engineering ontology.

The available experimentation process knowledge repository in eSEE includes processes to Experimental Study's Packaging and Execution [18], Experimental Study's Execution [13], Survey Planning and Execution Process [36] and Systematic Review Planning and Execution [9].

These experiences allowed us to observe that several Experimental Software Engineering process practices are similar to Software Engineering ones, such as documentation, training, configuration management, process management, and so on. Therefore, we believe that the experimentation process could be improved by incorporating some of the ISO/IEC 12207 processes. The objective is to include Software Engineering characteristics to the experimental process. For instance one of the experimental study *Planning* stage's sub-activities is *Results Validation Adequacy*. This sub-activity can be mapped into the ISO/IEC 12207's *Risk Management Process* (a sub-process of Management Process).

Besides supporting guidance to the execution of studies throughout its processes, eSEE also provides knowledge management facilities. The eSEE infrastructure must make available knowledge available regarding the experimentation process, methods,

techniques, and tools to assist the software engineering researcher while managing the study accomplishment. It allows lessons learned to be captured and stored for future reuse in the next studies' planning activities, both to prevent errors and to identify opportunities for improvements in the experimentation process.

6. Exploring Experimentation in eSEE

The previous sections have described the general features provided by eSEE to support researchers in the execution of studies in software engineering. Besides supporting individual primary and secondary studies, eSEE also allows to the researcher complementary possibilities to deal with these studies.

To develop new software technologies based on experimentation eSEE environment offers an extended experimental methodology for introducing software processes based on Shull et al. [44] through the adoption of systematic reviews (Figure 3) [34]. In this sense, the resulting methodology is divided into two steps: Technology Definition and Technology Refinement.

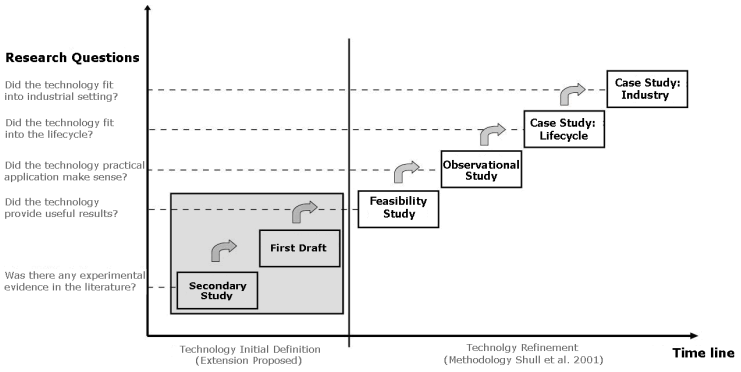

Figure 3. eSEE Experimentation Software Technology Development Model [34]

The first step is represented by the Initial Definition of Technology, in which a systematic review would be undertaken. The systematic review planning, as represented by the protocol definition, would provide support to delimitate the scope of the research. The presence of such a protocol would support evolving the problem under investigation, since it explicitly defines research questions, source selection, and both acceptance and quality criteria to be observed in those studies.

Furthermore, the procedures to identify and characterize the existing evidence on the subject can hardly reduce the underlying risks associated with the definition of a new technology. As a result of such characterization, improvement opportunities could be identified. In addition, such characterization could also minimize the possibility of repeating mistakes that might have been previously made.

The researcher defines an initial proposal based both on the knowledge gathered and on the evidences identified throughout the accomplished systematic review. Once a first technology version is done, the steps related to the series of experimental studies proposed in the original methodology are performed, in order to refine the technology. Examples of using such experimental methodology to develop software technologies can be found in [17,28,34].

7. Conclusions

Software Engineering still makes little usage of scientific methods during the definition of new software technologies, which are frequently defined in laboratories without a proper process for industry transfer. The lack of characterization of a software technology in use, due to the lack of a formal process to this end, makes Software Engineering a room for speculations on the quality and efficacy of new software technologies [27].

By adopting an evidence based approach, characterized by the conduction of primary and secondary studies, Software Engineering could reach a high quality degree. In this way, as pointed out by Juristo and Moreno [27], it would be possible to evolve from a speculation based software development to a fact based software development. Such evolution would allow transforming the process of building software in a predictable process. In this way, maybe we are going to be more experimental and less trial and error prone in the Software Engineering field [50].

The expected contributions of this research line can be represented by the following results:

- Making available a set of software components that can compose a computerized infrastructure to allow instantiation of experimentation environments for different experimental study types in Software Engineering;
- Reducing efforts and consequently the time for defining, planning, analyzing and packaging experimental studies in Software Engineering;
- Making available knowledge and associated tools that can be used by the academia and software industry to improve the development of their software technologies;

- Allowing the replication of experimental studies, that can support the improvement of the Software Engineering body of knowledge and;
- Extending the experimentation processes set by mapping ISO 12207's processes into the experimental life cycle.
- Adapting the experimentation process regarding the four staged taxonomy [52] and the definition of a standard experimentation process to support each one of them;
- Developing an experimentation ontology definition to organize knowledge about experimental software engineering in eSEE and to identify general characteristics of each experimental study type; and
- Defining an initial set of eCASE tools to populate the eSEE infrastructure.

We hope that Software Engineering and other areas researchers would benefit from using a computerized infrastructure, such as the eSEE, to support their experimentation processes and to package studies results.

6. Acknowledgments

eSEE is part of the Experimental Software Engineering and Science in Large Scale Project CNPq (475459/2007-5) and FAPERJ. The authors would like to thank the ESE Group work (http://www.cos.ufrj.br/~ese) for its valuable contribution on the building of eSEE and Prof. José Carlos Maldonado for his valuable suggestions regarding this paper.

7. References

[1] Antman E., Lau, J., Kupelnick, B., Mosteller, F., Chalmers, T. (1992) "A comparison of results of meta-analysis of randomized controlled trials and recommendations of clinical experts", JAMA, 268(2):240-248, July 1992.

[2] Arisholm, E., Sjøberg, D.I., Carelius, G.J., Lindsjørn, Y. (2002). "A Webbased Support Environment for Software Engineering Experiments", In: Nordic Journal of Computing, v. 9, n. 3 (September), pp. 231-247.

[3] Barcelos, R.F., Travassos, G.H. (2006): Evaluation Approaches for Software Architectural Documents: a Systematic Review. In: IDEAS 2006. Argentina

[4] Basili, V.R., Selby, R.W., Hutchens, D.H. (1986) "Experimentation in Software Engineering". In: IEEE TSE, 12 (7), pp.1728-1298.

[5] Basili, V.R., Shull, F., Lanubile, F. (1999). "Building Knowledge through Families of Experiments", In: IEEE TSE, vol. 25, No. 4.

[6] Basili V., Rombach D., Schneider K., Kitchenham B., Pfahl D., Selby R. (2006). Empirical Software Engineering Issues. Critical Assessments and Future Directions. International Workshop. LNCS, Vol. 4336. ISBN 3-540-71300-5.

[7] Basili, V.R. (2006) – "The Past, Present and Future of Experimental Software Engineering", In: JBCS – Journal of the Brazilian Computer Society. no. 3; vol. 12; Dez. - ISSN 0104-6500

[8] Basili, V.R, Zelkowitz, M.V. (2007). "Empirical Studies to Build a Science of Computer Science". CACM. Vol. 50. no 11. pp 33-37.

[9] Biolchini, J., Mian, P.G., Natali, A.C., and Travassos, G.H. (2005). "Systematic Review in Software Engineering: Relevance and Utility." Technical Report. PESC - COPPE/UFRJ. Brazil. Available at: http://cronos.cos.ufrj.br/publicacoes/reltec/es67905.pdf

[10] Biolchini, J. ; Mian, P. ; Conte, T. U. ; Natali, A.C.C. ; Travassos, G. H. (2007) . "Scientific research ontology to support systematic review in Software Engineering". Advanced Engineering Informatics, v. 21, p. 133-151.

[11] Cavalcanti, M.C., Baião, F., Rössle, C., Bisch, P.M, Targino, R., Pires, P.F., Campos, M.L., Mattoso, M. (2003). Strucutural Genomic Workflows Supported by Web Services. In: BIDM'03. In conjunction with DEXA 2003, Prague, Czech Republic.

[12] Chapetta, W. A., Santos, P.S.M., Travassos, G. H., (2005). "Supporting Meta- Description Activities in Experimental Software Engineering Environments". In: ESELAW'05, Brazil.

[13] Chapetta, W. A. An infrastruture for Planning, Execution and Packing Experimental Studies in Software Engeneering. Master Theses (in portuguese) PESC/COPPE -UFRJ, Rio de Janeiro, Brazil, 2006.

[14] Cochrane Collaboration (2003), Cochrane Reviewers' Handbook. Version 4.2.1. http://www.cochrane.org/resources/handbook/index.htm, last access in 08/01/2008.

[15] Cochrane, Al. (1989) In Chalmers I, Enkin M, Keirse MJNC, eds. "Effective care in pregnancy and childbirth". Oxford University Press, Oxford.

[16] Conte, T., Mendes E., Travassos, G. H. (2004). "Web Application Development Processes: a Systematic Review" (in Portuguese). In: WEBMEDIA. Brazil

[17] Conte, T., Massolar, J., Mendes, E., Travassos, G.H. (2007). "Usability Evaluation Based on Web Design Perspectives". In: IEEE/ACM 1st ESEM. pp. 146-155

[18] Costa, H.R.; Mian, P.G., Travassos, G.H. (2004). Extending a Process and packaging model for Experimental Studies. Techinical Report (in portuguese).

[19] Coutinho, A.L.G.A, Evsukoff, A.G., Werner, C.M.L., Travassos, G.H., Alves, J.L.D., Landau, L., ttoso, M.L.Q., ecken, N.F.F. (2006). "An Integrated Computational Environment for Modeling, Simulation and Visualization of Sedimentary Basins and Petroleum Systems". XXIX CNMAC. pp 517-518. Brazil.

[20] Dias Neto, A.C., Barcelos, R., Chapetta, W.A., Santos, P.S.M., Mafra, S.N., Travassos, G.H. (2004), "Infrastructure for SE Experiments Definition and Planning". In: ESELAW'04, Brasília, Brazil.

[21] Dias Neto, A.C.; Subramanyan, R.; Vieira, M.; Travassos, G.H. (2007), "A Survey on Model-based Testing Approaches: A Systematic Review", In: WEASELTech'07, Atlanta, November.

[22] EBSE (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-001. Available at http://www.dur.ac.uk/ebse/Systematic-reviews-5-8.pdf. Last access in 08/01/2008

[23] Feitelson, D.G. (ed) (2006). Experimental Computer Science: Introduction. ". CACM. Vol. 50. no 11. pp. 24-59.
[24] Jasper, R., Uschold, M. A. (1999). Framework for Understanding and Classifying Ontology Applications. In: KAW'99, Alberta, Canada.
[25] Jedlitschka, A., Pfahl, D. (2005). "Reporting Guidelines for Controlled Experiments in Software Engineering", In: 4th IEEE/ACM ISESE.
[26] Juristo, N., Moreno, A. (2001). "Basics of Software Engineering Experimentation", Kluwer Academic Press, 1st edition.
[27] Juristo, N., Moreno, A. (2002) "Reliable Knowledge for Software Development", IEEE Software, pp. 98-99, sep-oct, 2002.
[28] Kalinowski, M., Travassos, G.H., 2005, "Software Technologies: The Use of Experimentation to Introduce ISPIS - a Software Inspection Framework - Into the Industry". In: ESELAW'05, Uberlândia-MG, Brazil.
[29] Kim, W., Graupner, S., Sahai, A., Lenkov, D., Chudasama, C., Whedbee, S., Luo, Y., Desai, B., Mullings, H., Wong., P.. (2002). "Web E-speak: facilitating Web-based e-Services". Multimedia, IEEE, v. 9, n. 1 (Jan-Mar), pp. 43-55.
[30] Kitchenham B. A., Dybå T and Jørgensen M. (2004). Evidence-Based Software Engineering. In: ICSE 2004, 273--281, IEEE Computer Society Press.
[31] Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Joint Technical Report Software Engineering Group, DCS Keele University, United Kingdom and Empirical Software Engineering, NICTA Australia.
[32] Mafra, S., Travassos, G.H. (2005): Software Reading Techniques: A Systematic Review. In: XIX SBES. Brazil. (in Portuguese).
[33] Mafra, S.; Travassos, G.H. (2006) "Estudos Primários e Secundários Apoiando a Busca por Evidência em Engenharia de Software". Technical Report ES-687/06, PESC/COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
[34] Mafra, S.N.; Barcelos, R.F.; Travassos, G.H. (2006). "Applying an Evidence Based Methodology to Define New Software Technologies". In: XX SBES. Brazil. v. 1. p. 239-254. (in Portuguese)
[35] Mendes, E. Kitchenham, B. (2004): Protocol for Systematic Review. Available at: http://www.cs.auckland.ac.nz/emilia/srspp.pdf. Last accessed by 7/1/2008.
[36] Mendonça, C.C. A web support infrastruture for Surveys Planning and Execution and Packing. Master Theses (in portuguese) PESC/COPPE - UFRJ, Rio de Janeiro, Brazil, 2005.
[37] Mian, P.G.; Travassos, G.H.; Rocha, A.R.C. and Natali, A.C.C. (2004). "Towards a Computerized Infrastructure for Managing Experimental Software Engineering Knowledge". In:JIISIC'04, Spain.
[38] Mian, P., Conte T., Natali, A., Biolchini, J., Mendes, E. and Travassos, G. (2005). "Lessons Learned on Applying Systematic Reviews to Software Engineering". In: WSESE2005. Finland.
[39] Mian, P., Conte, T., Natali, A., Biolchini, J., and Travassos, G. (2005). "A Systematic Review Process to Software Engineering". In: ESELAW'05. Brazil.
[40] Newman, H.B., Ellisman, M.H. and Orcutt, J.A. (2003). "Data-Intensive E-Science Frontier Research". CACM, November , vol. 46, no. 11.
[41] NHS Centre for Reviews and Dissemination. (2003), "Database of Abstracts of Reviews of Effectiveness". In: The Cochrane Library, Issue 1. Oxford: Updated quarterly.
[42] Pai, M., McCulloch, M., Gorman, J.D., Pai, N., Enanoria, W., Kennedy, G., Tharyan, P. Colford Jr, J.M. (2004). "Systematic Reviews and meta-analysis: An illustrated, step-by-step guide". The National Medical Journal of India. Vol 17, No 2
[43] Santos, P.S.M. and Travassos, G.H. (2007). "eSEE – Ambiente de Apoio a Experimentação em Larga Escala em Engenharia de Software", In: 1st Brazilian e-Science Workshop, João Pessoa, PB, Brazil, October.
[44] Shull, F., Carver, J., Travassos, G. (2001) "An Empirical Methodology for Introducing Software Processes", In: 8th ESEC/FSE-9, pp. 288-296.
[45] Shull, F.; Mendonça, M.; Basili, V.; Carver, J.; Maldonado, J. C.; Fabbri, S.; Travassos, G. H.; Ferreira, M. C. (2004). "Knowledge-Sharing Issues in Experimental Software Engineering". Empirical Software Engineering An International Journal, USA, v. 9, n. 1-2, p. 111-137
[46] Souza, A.B.O. (2007). "Configurando Modelos de Processos de Experimentação em Engenharia de Software". XXIX Jornada Giulio Massarani de Iniciação Científica, Artística e Cultural (JIC 2007), DCC/UFRJ, October, Rio de Janeiro, RJ, Brazil.
[47] Spínola, R.O., Silva, J.L.M., Travassos, G.H. (2007) "Checklist to Characterize Ubiquitous Software Projects". In: XXI SBES. Brazil.
[48] Thomke, S.H. (2003). Experimentation Matters: Unlocking the Potential of New Technologies for Innovation. HBS Press. ISBN 1-57851-750-8
[49] Tichy, W.F. (1998) "Should Computer Scientists Experiment More?" In: IEEE Computer, 31 (5), p. 32-39.
[50] Travassos, G.H.(2006). "From Silver Bullets to Philosophers Stones: Who wants to Be Just an Empiricist?" In: LNCS 4336 - Empirical Software Engineering Issues Critical Assessment and Future Directions. Berlin: Springer -Verlag, v. 4336, p. 39-39.
[51] Travassos, G. H., Silva, L. F. S., Spinola, R. O., Kalinowski, M., Chapetta, W. (2003) "Tools and Facilities for Experimentation: Can we get there?" In: Panel presentation on the 11th International Software Engineering Research Network Meeting (ISERN 2003), Italy.
[52] Travassos, G.H., Barros, M.O. (2003). "Contributions of *In Virtuo* and *In Silico* Experiments for the Future of Empirical Studies in Software Engineering". In:WSESE' 03, Fraunhofer IRB Verlag, Roma.
[53] Wöhlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A. (2000) "Experimentation in Software Engineering: An Introduction", The Kluwer International Series in Software Engineering, Norwell, USA, Kluwer Academic Publishers.
[54] Zelkowitz, M.V, Wallace, D.R. (1998). "Experimental Models for Validating Technology", In: IEEE Computer, 31 (5), p. 23-31.