Proceedings of
SEDES 2012
Fourth Portuguese Software Engineering Doctoral Symposium

Miguel Goulão, Fernando Brito e Abreu

(Eds.)

Book chapter in *Proceedings of the 8th International Conference on the Quality of Information and Communications Technology* (*QUATIC'2012*), João Pascoal Faria, Alberto Silva, Ricardo J. Machado (Eds.), IEEE Computer Society, ISBN 978-0-7695-4777-0

Published by:

CPS Conference Publishing Services

Foreword
SEDES 2012
Fourth Portuguese Software Engineering Doctoral Symposium

Miguel Goulão[1], Fernando Brito e Abreu[2,1]
[1] CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa, Caparica, Portugal
[2] Departamento de Ciências e Tecnologias da Informação, Escola de Tecnologias e Arquitectura,
Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
mgoul@fct.unl.pt, fba@iscte-iul.pt

I. INTRODUCTION

THE Fourth edition of the Portuguese Software Engineering Doctoral Symposium (SEDES 2012) was held on 3 September 2012 in Lisbon, co-located with the QUATIC 2012 conference. The aim of this symposium is to bring together software engineering PhD students supervised or co-supervised by faculty members of Portuguese universities in a constructive environment where they can present and discuss their ongoing PhD projects. The symposium aims at selecting PhD students that have already settled on a specific research topic, but are at least one year apart from delivering their dissertation, so that they can still benefit from the symposium discussions. This symposium is also instrumental as a gathering point for the Software Engineering researchers in Portugal, with a tradition of synergies facilitator since its first edition. As such, an effort has been made to hold this event in different towns. Previous editions took place in Coimbra [1], Caparica [2] and Porto [3].

Regarding scope, the eligible topics for participation in SEDES include all knowledge areas defined in SWEBOK [4]. Research work in related fields such as Computer Science (including formal methods) and Information Systems (including Information Services variants) are considered within the scope of SEDES if they hold an applied perspective on the technological or methodological issues of software development or maintenance.

II. ORGANIZATION

Six PhD students were selected for presenting their work in this edition. Submissions were blind-reviewed by at least two program committee members focusing on the quality, maturity and clarity of the ongoing research work, both in terms of scope delimitation and problem relevance, adequacy of the adopted methodology, results significance and their validation, as well as technical writing style. The reviewers' panel included Software Engineering experts from the majority of the Portuguese public universities, namely Dulce Domingos (FCUL), João Cachopo (IST-UTL), João Miguel Fernandes (Univ. do Minho), José Maria Fernandes (Univ. de Aveiro), José Paulo Leal (FCUP), Miguel Pessoa Monteiro (FCT-UNL), Pedro Guerreiro (Univ. do Algarve), Raul Moreira Vidal (FEUP), and João Varajão (UTAD). The papers presented in the symposium covered a wide range of topics and were discussed in depth, both with the symposium participants and with a set of invited senior "opponents", chosen amongst SEDES PC and Steering Committee members.

The Symposium was organized in 4 sessions. The first session included a welcome address and an "elevator pitch session", where students were invited to present their work in no more than 2 minutes each. The session continued with the first student presentation. The second and third sessions included 2 student presentations each, while the fourth session featured the last student's presentation and a closing discussion. While the invited senior "opponents" were responsible for fostering a constructive discussion on the challenges faced by each of the students, all symposium participants (and, in particular, the PhD students) were strongly encouraged to provide feedback to the presenters.

III. PRESENTATIONS SUMMARY

Luís Alves presented his work on *"Experimental Software Engineering in Educational Context"*. In his PhD research, Luís is dealing with the challenges of conducting experimentation using students as participants. Experimentation is a crucial activity in the evaluation of Software Engineering claims. Unfortunately, it is hard to find appropriate industry settings where experimentation is feasible. Luís is particularly interested in evaluating how the Rational Unified Process can be made compliant with the Capability Maturity Model Integration (CMMI) maturity levels 2 and 3. He is also analyzing the influence of project management tools in the evolution of the maturity of development teams. By conducting his experimentation with students, Luís will assess the extent to which the results obtained with students are comparable with those reported in related studies carried out in industry.

Ankica Barišić presented her work on *"Usability Evaluation of Domain Specific Languages"* (DSLs). DSLs are increasingly being adopted in industry due to their claimed benefits with respect to software development productivity. These languages use concepts from the corresponding application domain, thus making them potentially suitable for usage not only by professional software developers, but also by domain experts and domain users, who can then develop their own applications. In order for DSLs to succeed, their quality in use is a key element. However, there is currently little evidence of serious DSL evaluation being carried out in a systematic way. Ankica's work aims to mitigate this shortcoming by proposing methods and models to support this evaluation and promote it to a first class activity in the DSL development cycle. This work draws influences from usability engineering to language engineering.

Tiago Boldt Sousa presented his work on *"Object-Functional Patterns: Re-Thinking Development in a Post-Functional World"*. For several years, design patterns have been primarily presented using the object-oriented paradigm. More recently, the development of other paradigms has provided developers with innovative ways of solving problems. Tiago focuses his work in the increasingly popular object-functional paradigm and on how existing patterns can be migrated to this paradigm and improved by using its mechanisms. The dissertation's contribution includes reference implementations for these patterns in the *Scala* programming language, to be evaluated both in academic and industrial contexts. The benefits of patterns introduction in this new paradigm are expected to be a valuable input for language development, through patterns absorption, as well as useful for practitioners who can apply these patterns in their work.

José Martins presented his work on *"Ontologies for Product and Process Traceability at Manufacturing Organizations: A Software Requirements Approach"*. A traceability business process is a mandatory feature for organizations acting as product providers, but its implementation in a sustainable way remains a challenge, mostly due to difficulties in reaching a common understanding on the meaning of traceability concepts, concrete demands and the process nature itself. José's work aims to improve the support for traceability offered by Information Systems solutions. To this end, José is developing an ontology of the "traceability business process", upon which domain models can then be built. The main targets (from a software development perspective) are requirements elicitation and solution validation. The reported work is being conducted in close cooperation with a large manufacturing organization partner, which will foster its validation in a real-world setting.

Manuel Amaro presented is work on *"A Software Framework for Supporting Ubiquitous Business Processes: An ANSI/ISA-95 Approach"*. Ubiquitous computing is becoming increasingly important, and has a potentially deep impact in the way business processes are shaped and monitored. In particular, monitoring the execution of business processes in real-time, through ubiquitous computing, enables the possibility of adapting these business processes to changes in their environment, as well as to set up alarms to detect deviations to the planned business processes (e.g. time deviations). This approach has already been tested in two projects in the automotive industry and is currently undergoing a formalization initiative, which will facilitate the construction of a framework to monitor the real-time executions of ubiquitous business properties.

Finally, José Sousa presented his work on *"Modeling Organizational Information Systems Using "Complex Networks" Concepts"*. His work tackles the problem of understanding the information flows that emerge in the increasingly more frequent Service Oriented Architecture (SOA) information systems. José is particularly interested in defining an approach to support the adoption of a complex network metamodel upon which existing organizational information systems can be defined and later monitored and better understood. The work is inspired by complex networks research from other domains, namely physics and is expected to be instrumental in increasing our understanding on the co-evolution of enterprise and socio-technical systems.

IV. Conclusions

Once again, SEDES was a privileged occasion for Software Engineering PhD students to get feedback on their research proposals and on the adequacy and feasibility of their research plans, as well as for getting advice on how to improve their scientific presentation abilities.

Further information on this doctoral symposium can be found at http://2012.quatic.org/sedes/

ACKNOWLEDGMENT

The SEDES 2012 organizing chairs thank QUATIC 2012 organizing chairs for their valuable support in hosting this SEDES edition.

REFERENCES

[1] *SEDES'2004*, http://ctp.di.fct.unl.pt/QUASAR/sedes2004/, Fernando Brito e Abreu (organizer), hosted by the *1º Congresso Português de Engenharia de Software*, Coimbra, Portugal, April 2004.

[2] *SEDES'2007*, http://quatic2007.dsi.uminho.pt/workshops.html#SEDES, Paulo Rupino da Cunha (organizer), hosted by the *6th International Conference on the Quality of Information and Communication Technology (QUATIC'2007)*, Lisbon, Portugal, September 2007.

[3] *SEDES'2009*, http://www.iaria.org/conferences2009/SEDES.html, Ricardo J. Machado and João M. Fernandes (organizers), hosted by the *4rd International Conference on Software Engineering Advances (ICSEA'2009)*, Porto, September 2009.

[4] *SWEBOK: Guide to the Software Engineering Body of Knowledge,* http://www.swebok.org, Alain Abran, James W. Moore, Pierre Bourque, Robert Dupuis (eds), IEEE Computer Society, 2004.

Experimental Software Engineering in Educational Context

Luís M. Alves
Escola de Tecnologia e Gestão
Instituto Politécnico de Bragança
Bragança, Portugal
lalves@ipb.pt

Ricardo J. Machado
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
rmac@dsi.uminho.pt

Pedro Ribeiro
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
pmgar@dsi.uminho.pt

Abstract — **Empirical studies are important in software engineering to evaluate new tools, techniques, methods and technologies in a structured way before they are introduced in the industrial (real) software process. Within this PhD thesis we will develop a framework of a consistent process for involving students as subjects of empirical studies of software engineering. In concrete, our experiences with software development teams composed of students will analyze how RUP (Rational Unified Process) processes can be compliant with the CMMI (Capability Maturity Model Integration), namely in the context of MLs (maturity levels) 2 and 3. Additionally, we will also analyze the influence of project management tools to improve the process maturity of the teams. Our final goal of carrying out empirical studies with students is to understand its validity when compared with the corresponding studies in real industrial settings.**

Keywords: software engineering management, software engineering process, software quality

I. Introduction

In the early nineties, Basili introduced, for the first time, the concept of *experience factory*. As the author refers in [1] the concept was introduced to "institutionalize the collective learning of the organization that is at the root of continual improvement and competitive advantage". Thus, the *experience factory* provides an organizational schema for collecting experiences on reuse of empirical results, for analyzing them and generalizing the knowledge contained [2]. This scheme was designed based on many years of the Software Engineering Laboratory (SEL) work. Over several years, this well-known laboratory has conducted several studies and experiments for the purpose of understanding, assessing, and improving software and software processes within a production software development environment at the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) [1].

With our approach we do not intend to create a new software engineering laboratory. Instead, we intend to create a space (virtual or physical) that allows us to conduct empirical studies in the software engineering area by involving students that are enrolled in our current software engineering courses (both at undergraduate and postgraduate university programmes).

Unlike other mature disciplines, the field of software engineering continues to lack a research and development infrastructure that supports systematic testing of novel software engineering methodologies. Our intention is to develop a new experience factory approach based on one explicit educational environment. Initially, we will work just with students as subjects of our first empirical studies. We are fully aware that we will face some problems with the validation of the results that we will be obtained in our student-based experiments. It is impossible to be sure that techniques evaluated under such circumstances will scale up to industrial size systems or very novel software engineering problems. Even though, Kitchenham says that "students are the next generation of software professionals and, so, are relatively close to the population of interest" [3]. In the opposite, students in psychology studies are not representatives of the human population as a whole [4].

In this paper, a description of the state-of-the-art related with the subject of this research is presented in Section 2. Section 3 describes in detail the research objectives and the methodological approach. In Section 4, the past work and preliminary results already done in the context of this research are briefly described. Section 5 presents the future work and expected results for the next 2 years of research. Finally, in Section 6 some conclusions are presented

II. STATE-OF-THE-ART

The state-of-art of this work essentially relates to: ESE (Empirical Software Engineering), SPI (Software Process Improvement) and PM (Project Management). We will give special emphasis to the ESE with students as subjects of experiments.

A. Empirical Software Engineering

ESE is a sub-field of software engineering which aims at applying empirical theories and methods for the measuring, understanding, and improvement of the software development process in real software companies [5]. This definition extends the concept for ESE proposed by Basili, when he said that "experimentation is performed in order to help us better evaluate, predict, understand, control, and improve the software development process and product" [6]. In the early nineties, the empirical methods applied in software engineering were basically restricted to quantitative studies (mostly controlled experiments). The concept of experimental software engineering has moved to empirical software engineering when a range of qualitative methods have been introduced, from observational to ethnographical studies. In a broad sense, an empirical investigation (synonym of empirical study) is a process that aims to discover something unknown or to validate hypotheses that can be transformed in generally valid laws [2].

It is important to be able to evaluate new techniques and methods in a structured way before they are introduced in the software process [7]. Empirical methods have gained increased attention in software engineering; there are dedicated conferences such as the International Conference on Evaluation and Assessment in Software (EASE), and there are dedicated journals such as the International Journal of Empirical Software Engineering.

Controlled experiments are the most commonly used empirical methods in software engineering. Sjøberg *et al.* define controlled experiment in software engineering as a "randomized experiment or a quasi-experiment in which individuals or teams (the experimental units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages, or tools (the treatments)" [8]. Sjøberg *et al.* analyzed in detail 103 scientific articles published in leading software engineering journals and conferences in the decade from 1993 to 2002 that reported controlled experiments in which individuals or teams performed one or more software engineering tasks.

Currently, some universities offer courses in the ESE area, as in the cases of Norwegian University of Science and Technology [5] and Lund University in Sweden. Both institutions have worked with students as subjects of experiments. These institutions run the experiences out of the courses' context, whereas in our approach the students perform the experiments as part of their regular academic courses. The Department of Computer Science of the University of Helsinki created an experimental software laboratory for basic and applied software development research and education. The name of this laboratory is *Software Factory* and they involve researchers, students, and industry partners in their projects [9].

B. ESE using Students versus Profissionals

In this section, based on literature review we will describe the strengths/weaknesses of using students versus professionals in the empirical software engineering context.

In the survey conducted in [8], a total of 5,488 subjects took part in the 113 experiments investigated, eighty-seven percent were students and nine percent were professionals. This survey demonstrates the importance of using students in this context.

In many studies, students are used instead of professional software developers, although the objective is to draw conclusions valid for professional software developers. The differences are only minor, and it is concluded that software engineering students may be used instead of professional software developers under certain conditions. Höst *et al.* [10] argue that the main reason to use students as subjects is often that they are available at universities and they are willing to participate in studies as part of courses they attend. In many cases, it is possible to combine the learning objectives of the courses with the research objectives of the studies. Tichy refer that software students are much closer to the world of software professionals than psychology students are to the general population [11]. In particular, software graduate students are so close to professional status that the differences are marginal. Software graduate students are technically more up to date than the "average" software developer who may not even have a degree in computing. Software professionals, on the other hand, may be better prepared in the application domain and may have learnt to deal with systems and organizations of larger scale than a student.

Sjøberg *et al.* [12] argue that the main reason of most subjects in software engineering experiments are students is that they are more accessible and easier to organize, and hiring them is generally inexpensive. Consequently, it is easier to run an experiment with students than with professionals and the risks are low. Jaccheri [13] refers that empirical studies are often carried out with students because they are viewed as inexpensive subjects for pilot studies. Svahnberg [14] refers that the students are readily available, often willing to participate, and require no or little compensation. The bad thing is that the variations among studies conducted with professionals are higher than the variations among students due to the more varied educational backgrounds and working experiences in the professionals [12].

Carver *et al.* [15] have developed a checklist that provides guidance for researchers and educators when planning and conducting studies in university courses. In our PhD work, we want to specialize this framework to the software engineering domain, when conducting experiences related with software process improvement and project management research questions.

C. Software Process Improvement

According to Humphrey [16], a software process is "the sequence of steps required to develop or maintain software, aiming at providing the technical and management framework for applying methods, tools, and people to the software task". Therefore, SPI aims at providing software development companies with mechanisms for evaluating their existing processes, identifying possibilities for improving as well as implementing and evaluating the impact of improvements [17].

SPI is an applied academic field, rooted in the software engineering and information systems disciplines, which has been studied for almost twenty years now. It deals primarily with the professional management of software companies, and the improvement of their practice, displaying a managerial focus rather than dealing directly with the techniques that are used to write software. Classical SPI techniques relate to software processes, standardization, software metrics, and process improvement. Many of the major contributions to SPI are originated from the SEI (Software Engineering Institute) at Carnegie Mellon University [18] [36].

SPI is based on process assessment. Most process improvement models and standards applied in SPI primarily provide guidance for process assessment. When critical-mission software is required to demonstrate (often by obtaining certain type of certifications) their ability to develop and sustain high maturity practices is mandatory. There are currently some software process models available for assessing and improving software development and its related practices.

Empirical studies that we will perform during the PhD work will concentrate primarily on the software development process, from the perspective of process improvement. Thus, we intend to implement experiments involving the suggested practices in CMMI (Capability Maturity Model Integration) [19] and RUP (Rational Unified Process) [20].

D. Project Management Approaches

One of the standard models most popular in PM area is the PMBOK (Project Management Body of Knowledge) [21]. Thus, in 1996, the first version of the body of knowledge in PM was published by the Project Management Institute [22]. According to the PMBOK, projects are composed of processes. A process is "a set of interrelated actions and activities performed to achieve a pre-specified product, result or service. Each process is characterized by its inputs, the tools and techniques that can be applied, and the resulting outputs" [21].

Today, one can find several approaches that aim at collecting PM data in a standardized data model which can be used to implement PM tools and to exchange project data. In order to perform PM activities, people use different methodologies according to their needs and standards. Instead of creating a project plan manually, companies use PM tools that support most important PM processes [21]. For instance, *Microsoft Office Project* is one of the most often used PM tools in small teams [34]. Although it is not based on an official standard, it can surely be considered as a de-facto standard because of its market position. However, this tool does not have an open structure since it uses a proprietary data model, which is not defined by an independent body.

PROMONT [35] is an ontology-based PM approach that intends to summarize all major PM standards and tools in one integrated reference model. It offers extending definitions of PM issues aimed at supporting interoperability of PM systems, processes and organizations. In particular, PROMONT offers a formal approach to define relationships and conditions between different terms that are used in PM.

III. RESEARCH OBJECTIVES AND METHODOLOGICAL APPROACH

A. Research objectives

It is common knowledge that software projects have a high rate of failure [23]. Although various strategies have been tried (such as structured programming, rapid prototyping, CASE tools and so forth), there is still no end to the software crisis.

With the intensification, acceleration in the rate of change, and expansion in the use of information technologies, particular attention is being focused on the opportunities and difficulties associated with sharing knowledge and transferring "best practices" within and across organizations [24]. A best practice is public knowledge, a tactic or method that has been shown through real-life implementation to be successful [25]. Models and standards that provide guidance for process improvement include a set of best practices for product and service development and maintenance [19].

A typical problem with software engineering research is that either it is difficult to find companies that provide reasonable research possibilities or the research is made with students in "artificial environments". Our approach provides a solution for this problem. In our approach we can do research in a very similar authentic environment. The participants in our experiments are students but the environment is very business-like. Teams work constantly together just like in a real work place. There is always a real business demand behind the project, which makes the project context valid for research. Researcher can observe team members anytime and even participate in projects if it is considered useful. Face to what we could allow in real company, our approach has some advantages, namely:

- The ease of research to use their own means of investigation and, at any time, the ease of the researcher to ask participants to answer questionnaires (paper or web) during the semester (within the classes or outside classes);
- All artifacts and documents (e.g. code, models and reports) provided by the teams are available for research purposes (we adopt direct analysis of artifacts to assess the teams process and product maturity);
- Researcher can go to the laboratory and do direct observation (teams have mandatory meetings in our laboratories and are available to be observed when interacting and working in their projects);
- Researcher can take part in the projects and interview both team members and clients during and after the projects.

This PhD thesis will adopt four main objectives. The first three correspond to specific software process research questions that are perfectly pertinent to be addressed when considering the configuration of process frameworks and PM tools in small software development teams. The fourth objective is related with the ESE perspective to assess empirical results with students; which means that efforts relative to this fourth objective must run in parallel with the others. The efforts relative to the first three objectives may not necessarily be run in a sequential order; we will adopt spiral approach to deal with the complexity of managing the complexity relative to all the existing interdependencies between the variables under study in the first three objectives:

- The first objective is to analyze the coverage of CMMI practices that we can expect when adopting the RUP reference model. To fulfill this objective, we need an alignment between CMMI and RUP process frameworks, by selecting the process areas, the specific goals and the specific practices from CMMI and comparing them with the coverage we can expect from the execution of the activities and tasks established by RUP.
- The second objective is to evaluate how CMMI ML2 and ML3 can be accomplished by particular configurations of RUP for small software development teams. To fulfill this

objective, we need to address the specific configurations of RUP and understand the implications in the alignment established in the pursuing of the previous objective. The outcome of these two first objectives may explain how to adopt RUP as a process asset to promote CMMI assessments, taking into account the specific characteristics of the team's organization (roles, tasks, activities).

- The third objective is to assess the impact of PM tools in the performance of software development teams. With this objective we intend to determine the relationship between the maturity of the teams and the support they can get from PM tools. The outcome of this third objective may explain what kind of key success factors we should look for when choosing one PM tool taking into account the process framework (in our case, configurations of RUP for small teams) and the maturity assessment reference model (in our case, CMMI ML2 and ML3) we adopt to frame the software development team.
- Finally, the last and most important objective is to validate the research results to be produced by the previous three objectives in an explicit educational context. The external validity is a major concern in the ESE. The external validity defines the conditions that limit the ability to generalize the results of an experiment to industrial practice. Problems can occur due to the population of participants not be representative of the population under interest, instrumentation is not suitable for industrial practice, and the experiment can be run in a day or special time that will affect the results. In our case, we will run three sets of experiences with students, each one dedicated to one the objectives previously referred. This fourth objective corresponds to an umbrella research question that will enable the production of some systematic insight of the advantages and drawbacks of conducting empirical studies with software students.

B. *Methodological approach*

An experience should be treated as a process of formulation or verification of a theory. In order that the process provides valid results, it must be properly organized and controlled, or, at least, monitored. In order to achieve these goals several methods of organization of experiments have been proposed. In order to compare the experimentation methodologies we have to consider their different characteristics, for example, the phases of process experimentation, the way of the transformation of abstract concepts of the domain to concrete metrics, the main purpose of experimentation, tools, etc.

In the sub-field ESE, the most relevant research methods are the controlled experiments, the surveys, and the case studies. The selection of methods for a given research project depends on many local contingencies, including available resources, access to subjects, opportunity to control the variables of interest, and, of course, the skills of the researcher [26]. All the research methods have known flaws and each can only provide limited, qualified evidence about the phenomena being studied. However, each method is flawed differently and viable research strategies use multiple methods, chosen in such a way that the weaknesses of each method are addressed by use of complementary methods [27].

We will adopt surveys as one of the research methods (specifically, questionnaires) since it is an assessment tool that can be applied to a considerable number of students, it is cost effective and non-invasive, provide quantitative data, and allows the analysis of results with promptness. It has been argued that the application of questionnaires consumes less time, effort and financial resources than other methods of data collection such as interviews and document reviews [28]. However, at later stages of the research, we will make some interviews with some students to get additional information about the team's organization (mainly related with the instantiation of RUP configurations).

State-of-the-art will be performed as another research approach at initial stages of the PhD work. This activity will complement the brief state-of-the-art presented in this paper. With the literature review, we intend to acquire knowledge about the efforts made for similar problems. We intend to review the following main areas of study:

- Experimental software engineering giving special attention to studies conducted with students as subjects;
- Software process improvement approaches, in particular CMMI and RUP configurations for small teams;
- Project management tools and their support to software development activities.

The three sets of experiences with students will be run as empirical software engineering studies, framed by all the recommendations contained in the previously referred literature. Simultaneously, with the validation of the research results, we will start the development of a framework that shows us a consistent process of using students as subjects of empirical studies. The writing of the thesis will be done along the realization of the work.

IV. PAST WORK AND PRELIMINARY RESULTS

This PhD work takes place within the Software Engineering and Management Group (SEMAG) from the ALGORITMI Research Centre at the University of Minho. SEMAG research group is devoted to study the development process of software-based information systems and related methodologies, focusing on both the engineering and management aspects.

At the undergraduate level (Bologna 1st cycle), the teaching staff of the SEMAG is mainly enrolled in the University of Minho DLic degree in Information Systems and Technology (LTSI) by running, among others, the Software Process and Methodologies (PMS) and Development of Software Applications (DAI) courses. At the postgraduate level (Bologna 2nd cycle), the teaching staff of the SEMAG is enrolled both in the DEng degree in Engineering and Management of Information Systems (MEGSI) and in the MSc degree in Information Systems (MSI) by running, among others, the Analysis and Design of Information Systems (ACSI) and Project Management for Information Systems (GPSI). The empirical studies planned for this PhD work will use software engineering materials and students from PMS, ACSI, DAI, and GPSI courses.

During the first academic semester, PMS students (undergraduation) perform part of the RUP inception phase relative to one real software application, resulting in a project proposal to be addressed to one real client. They have three moments of evaluation and their work focuses on business modeling, requirement, and project management disciplines. The existence of a real costumer permits the acquiring of all the needed information to perform the project proposal. Simultaneously, some ACSI students (postgraduation) get involved with PMS students in order to collect information about the produced business and requirements artifacts and to perform CMMI assessments.

In the second academic semester, DAI students (undergraduation) continue to serve the same client of the first semester and perform the remainder of the RUP inception phase and execute the elaboration, construction and transition phases of RUP to deploy the software application to the real client. Simultaneously, some GPSI students (postgraduation) get involved with DAI students to collect information about the produced software artifacts and the adopted RUP configuration and to perform CMMI assessments and to analyze the utilization of PM tools.

In our approach, we detain several mechanisms that bring into the educational context some characteristics of a real industrial project:

- We have a real client that interacts with the teams and that opens for them the real organizational environment where the software application will be explored;
- We adopt a real problem, with the complexity and the imperfections of any real medium-size software project;
- The inter-relation between PMS and ACSI courses (by means of the ACSI students that emulate external process consultants) and between DAI and GPSI courses (by means of the GPSI students that emulate senior project facilitators) allow us to recreate a typical industrial environment where we have outsourcing of consultants and several depths of professional experiences in the teams;
- The teams compete with each other to sell their software application to the client, which emulates reasonably well the real software market.

The two sets of undergraduate and postgraduate courses (PMS+ACSI and DAI+GPSI) allow us to perform empirical studies of the controlled experiment type, where teams of students (subjects) are the experimental units that lead several software engineering tasks to assess different software processes (RUP configurations) and PM tools support.

In the academic year of 2010/2011, a controlled experiment was performed to assess the reduced model of RUP [29] [30]. It involved seven development software teams. The teams had between 13 and 17 students (1 team with 13, 3 teams with 14, 2 teams with 16 and 1 with 17). Two teams (team 5 and team 7) were randomly chosen to not adopting the RUP reduced model (we called these two teams the "Control Teams"), while the other five teams followed the guidelines established by the RUP reduced model, executing the phases of inception, elaboration and construction. The students elaborated the project proposals during the first semester and developed the software applications during the second semester.

The assessment of the RUP reduced model was conducted by adopting the CMMI-DEV v1.2 ML 2 reference model. With the exception of SAM (Supplier Agreement Management), all the other process areas were assessed. Figure 1 shows the percentage of accomplishment of all specific practices from all process area analyzed for each team. Although there is a significant difference between the various teams, the obtained results show that when the teams use the RUP reduced model they are able to accomplish CMMI ML2 adequately [31].

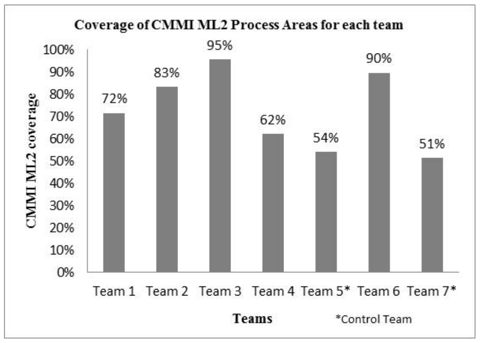

Figure 1. : Coverage of CMMI ML2 Process Areas

In this first experiment, students were suggested to use *Microsoft Project Server 2010* to support their software development activities. The configuration of this platform was performed by two GPSI students. The configuration was extremely difficult to perform. Teams had very little tool support to perform PM tasks.

In the academic year of 2011/2012, a second controlled experiment is being performed to assess the mapping between specific practices of CMMI ML2 and ML3 process areas and RUP artifacts, activities and tasks. In this second experiment, students are using *Clocking IT* [32] and *Teamwork Project Manager* [33] to support their software development activities. *ClockingIT* is an open source application hosted for tracking all tasks, issues, projects and time spent, with a focus on software development and handling large amounts of tasks. *Teamwork Project Manager* is an online application that helps organize and take control of our current projects, task lists, milestones, files, notebooks, resources and time. We intent to assess the influence of these tools in the team's performance. Meanwhile, we are gathering information to elaborate our framework to support the adoption of student teams to perform industry-valuable empirical software engineering experiences.

V. FUTURE WORK AND EXPECTED RESULTS

For the next two academic years (2012/13 and 2013/2014), students will get a more stable PM tool support. With the lessons learned from the two first experiments we intend to refine our processes of experimentation and start to explicitly address specific issues related with conceptual elaboration of our framework. We will also compare the CMMI maturity of teams that adopt the RUP reduced model with those adopting agile methods. We will also assess specific PM tools.

VI. CONCLUSIONS

Empirical studies in software engineering are important to be conducted to evaluate new tools, techniques, methods and technologies in a structured way before they are introduced in a real software process. Taking into account that: (1) software companies are not usually available to conduct empirical studies; and (2) when, exceptionally, they decide to do it, they keep the results for themselves; empirical studies with students are an interesting alternative to assess software processes and tools and share the results with the academia and the industry.

The problem with this interesting alternative is that there is a lack of scientific evidence that empirical studies with students are valuable for software companies. In our PhD work we intend to develop a framework that shows us a consistent process of using students as subjects of empirical studies. The framework will help to guide new empirical studies in a way that software companies may get interested in buying empirical studies to our laboratory. With this research we hope to contribute to the body of knowledge of ESE, SPI and PM and also to contribute to the increasing of the competitiveness of software companies.

REFERENCES

[1] V. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, *The Software Engineering Laboratory-an Operational Software Experience Factory*, in ICSE 92, pp. 370-381, 1992.

[2] A.D. Lucia, F. Ferrucci, G. Tortora, and M. Tucci, *Emerging Methods, Technologies and Process Management in Software Engineering*. John Wiley & Sons, 2008.

[3] 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*, in TSE, vol. 28, no. 8, pp. 721-734, 2002.

[4] R. Rosenthal, *Science and Ethics in Conducting, Analyzing, and Reporting Psychological Research*, in Psychological Sciense, vol. 5, no. 3, pp. 127-134, 1994.

[5] L. Jaccheri and T. Osterlie, *Can We Teach Empirical Software Engineering?*, in METRICS 2005.

[6] V. Basili, R.W. Selby, and D. H. Hutchens, *Experimentation in Software Engineering*, in TSE, vol. 12, no. 7, pp. 733-743, 1986.

[7] M. Höst, *Introducing Empirical Software Engineering Methods in Education*, in SEET 2002, pp. 170-179, 2002.

[8] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.K. Liborg, and A.C. Rekdal, *A Survey of Controlled Experiments in Software Engineering*, in TSE, vol. 31, no. 9, pp. 733-753, 2005.

[9] University of Helsinki. (2012, 2012-5-10). *Software Factory*. Available: http://www.softwarefactory.cc/

[10] M. Höst, B. Regnell, and C. Wohlin, *Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment*, in ESE, vol. 5, no. 3, pp. 201-214, 2000.

[11] W.F. Tichy, *Hints for Reviewing Empirical Work in Software Engineering*, in ESE, vol. 5, no. 4, pp. 309-312, 2000.

[12] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dyba, M. Jorgensen, A. Karahasanovic, E. F. Koren, and M. Vokac, *Conducting Realistic Experiments in Software Engineering*, in ISESE 2002.

[13] L. Jaccheri and S. Morasca, *Involving Industry Professionals in Empirical Studies with Students*, in ICESE 2007, Germany, 2007.

[14] M. Svahnberg, A. Aurum, and C. Wohlin, *Using Students as Subjects - An Empirical Evaluation*, in ESEM 2008.

[15] J.C. Carver, L. Jaccheri, S. Morasca, and F. Shull, *A Checklist for Integrating Student Empirical Studies with Research and Teaching Goals*, in ESE, vol. 15, no. 1, pp. 35-59, 2010.

[16] W. S. Humphrey, *A Discipline for Software Engineering*. Addison Wesley, 1995.

[17] W.A. Florac, A.D. Carleton, and J.R. Barnard, *Statistical Process Control: Analyzing Space Shuttle Onboard Software Process*, in IEEE Software, vol. 17, no. 4, pp. 97-106, 2000.

[18] B. Hansen, J. Rose, and G. Tjørnehøj, *Prescription, Description, Reflection: The Shape of the Software Process Improvement Field*, IJIM, vol. 24, no. 6, pp. 457-472, 2004.

[19] SEI, "*CMMI® for Development, Version 1.3*, Software Engineering Institute, CMU/SEI-2010-TR-033, 2010.

[20] P. Kruchten, *The Rational Unified Process - An Introduction*, 3rd Edition. Addison-Wesley, 2003.

[21] PMI, *A Guide to the Project Management Body of Knowledge*, Fourth Edition, Project Management Institute, 2008.

[22] PMI. Available: http://www.pmi.org

[23] The Standish Group. (2009). *Chaos Report*. Available: http://www1.standishgroup.com/newsroom/chaos_2009.php

[24] W.J. Orlikowski, *Knowing in Practice: Enacting a Collective Capability in Distributed Organizing*, Organization Sciense, vol. 13, no. 3, 249-273, 2002.

[25] R. G. Cooper, *Winning at New Products: Accelerating the Process from Idea to Launch*, third edition, Addison-Wesley, 2001.

[26] S. Easterbrook, J. Singer, M.A. Storey, and D. Damian, *Selecting Empirical Methods for Software Engineering Research*, in Guide to Advanced Empirical Software Engineering, 1st Ed., pp. 285-311, 2008.

[27] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 3rd edition,Sage Publications Inc., 2009.

[28] I. Garcia, C. Pacheco, and P. Sumano, *Use of Questionnaire-Based Appraisal to Improve the Software Acquisition Process in Small and Medium Enterprises*, in SERMA, vol. 150, pp. 15-27, 2008.

[29] P. Borges, P. Monteiro, and R. J. Machado, *Tailoring RUP to Small Software Development Teams*, in SEAA 2011, pp. 306-309, 2011.

[30] P. Borges, P. Monteiro, and R. J. Machado, *Mapping RUP Roles to Small Software Development Teams*, in SWQD 2011, pp. 59-70, 2012.

[31] F. Mandjam, *Avaliação do Impacto das Práticas do CMMI no Desempenho de Equipas de Desenvolvimento de Software no Ensino*, MSc in Engineering and Mangement of Information Systems, Universidade do Minho, Portugal, 2011.

[32] E. Simonsen and E. Simonsen. (2008, 2012-05-11). *Clocking IT TimeTracking 2.0*. Available: http://www.clockingit.com/

[33] Teamwork Project Manager. (2012, 2012-05-11). *Teamwork Project Manager*. Available: http://www.teamworkpm.net/

[34] Microsoft. (2012, 2012-05-10). Available: http://office.microsoft.com/pt-pt/project-help/ CH010362755.aspx

[35] S. Abels, F. Ahlemann, A. Hahn, K. Hausmann, and J. Strickmann, *PROMONT - A project management ontology as a reference for virtual project organizations*, in LNCS, vol. 4277, pp. 813-823, 2006.

[36] M.C. Paulk, "*A History of the Capability Maturity Model for Software*," ASQ Software Quality Professional, vol. 12, no. 1, pp. 5-19, 2009.

Usability Evaluation of Domain-Specific Languages

Ankica Barišić, Vasco Amaral, Miguel Goulão
CITI, Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
a.barisic@campus.fct.unl.pt, vma@fct.unl.pt, mgoul@fct.unl.pt

***Abstract*—Domain-Specific Languages (DSLs) are claimed to bring important productivity improvements to developers, when compared to General-Purpose Languages (GPLs). The increased Usability is regarded as one of the key benefits of DSLs when compared to GPLs, and has an important impact on the achieved productivity of the DSL users. So, it is essential to build in good usability while developing the DSL. The purpose of this proposal is to contribute to the systematic activity of Software Language Engineering by focusing on the issue of the Usability evaluation of DSLs. Usability evaluation is often skipped, relaxed, or at least omitted from papers reporting development of DSLs. We argue that a systematic approach based on User Interface experimental validation techniques should be used to assess the impact of new DSLs. For that purpose, we propose to merge common Usability evaluation processes with the DSL development process. In order to provide reliable metrics and tools we should reuse and identify good practices that exist in Human-Computer Interaction community.**

Keywords: Domain-Specific Languages, Usability Evaluation, Software Language Engineering

I. INTRODUCTION

An increasing number of people rely on software systems to perform their daily routines and responsibilities. As such, systems need to be developed rapidly. Domain-Specific Languages (DSLs) are claimed to contribute to a productivity increase in software systems development, while reducing the required maintenance and programming expertise. The main purpose of DSLs is to bridge the gap between the Problem Domain (crucial concepts, domain knowledge, techniques, and paradigms) and the Solution Domain (technical space, middleware, platforms and programming languages). The sooner we fill in this gap, the sooner we shall increase users' productivity. However intuitive this idea may be, we need to have means to assess the Quality and success of the developed languages. The alternative is to accept the risk of building inappropriate languages that could even decrease productivity or increase maintenance costs.

Software Language Engineering (SLE) is the application of a systematic, disciplined and quantifiable approach to the development, usage, and maintenance of software languages. One of the crucial steps in the construction of DSLs is their validation. However, this step is frequently neglected. The lack of systematic approaches to evaluation, and the lack of guidelines and a comprehensive set of tools may explain this shortcoming in the current state of practice. To assess the impact of new DSLs we could reuse experimental validation techniques designed for User Interfaces (UIs) evaluation. The focus of this research proposal is to build up a conceptual framework that supports the development process of DSLs concerning the Usability evaluation. This will include concepts, methods, languages, processes, implementation of tools, and metrics proposal.

DSLs can be regarded as communication interfaces between humans and computers. In that sense, using a DSL is a form of Human-Computer Interaction (HCI). As such, DSLs evaluation could benefit from techniques used for evaluating regular UIs. We reviewed current methodologies and tools for the evaluation of UIs and General Purpose Languages (GPLs), in order to identify opportunities for improving the current state of practice in DSL evaluation. That brought us closer to providing adequate techniques for supporting the evaluation process which, we argue, should be based on methods for assessing user experience and customer satisfaction, applied to DSL users. By promoting DSL Usability to a priority in the DSL development, Usability must be considered from the beginning of the development cycle. One way of doing this is through user-centered methods. In order to tailor such methods to DSL development, we need to establish formal correspondences for all stages of the DSL development process and the Usability evaluation process.

This paper is organized as follows. In section II we discuss the current state of the art in DSL development and potential contributions from HCI to improve it. In section III we detail our research objectives and methodology. In section IV we report on the preliminary results in this research project, while in section V we outline our plans for future work and expected results. In section VI we present the conclusions for this paper.

II. STATE-OF-THE-ART

The immersion of computer technology in a wide range of domains, leads to a situation where the users' needs become increasingly demanding and complex. The Quality of the users' interaction with this kind of technology is becoming of the utmost importance. Consequently, the development of successful software systems becomes increasingly more complex.

Software engineers need to cope with the growing of both essential and accidental complexity [1]. They have to provide solutions that solve a class of crucial problems in a given domain, which are sometimes very complex to learn, such as the rules and technical jargon found in domains like the Physics, Finance, Medicine, etc. Also, they need to deal with the accidental complexity of the used technology, *e.g.*, the use of low level abstraction programming languages, while integrating a wide plethora of different tools and libraries.

The use of the Model Driven Development (MDD) techniques and tools is seen as a viable approach for dealing with this accidental complexity[2]. MDD is grounded on the notion of providing explicit Models, commonly called *"first class artifacts"*, that are further translated into other lower level, more detailed, Models. These translations are also considered as development artifacts and can be explicitly modeled by means of transformation models. This approach has special impact in dealing with the complexity of large scale problems, while enabling rapid prototyping, simulation, validation and verification techniques [3], [4].

In direct relation with the MDD approach, we have modeling languages that are able to express the models with adequate notations. DSLs provide a notation tailored towards an application domain as they are based on models of relevant concepts and features of the domain [5]. As DSLs are used to describe and generate members of a family of systems in the application domain, they give the expressive power to generate the required family members more easily. As such they separate domain experts' work from analysis/transformation experts' work. DSLs are claimed to match users' mental model of the problem domain by constraining the user to the given problem [6].

In general, the software industry does not report investment on the evaluation of DSLs, as shown in a recent systematic literature review [7]. This conveys a perception that there is an insufficient understanding of the SLE process which, in our opinion, must include the evaluation of the produced DSLs. This apparent state of practice contrasts with the return of investment attributed to usability improvements reported for other software products [8]. In general, those benefits span from a reduction of development and maintenance costs, to increased revenues brought by an improved productivity by the end users [9].

The end user of the DSL can be a domain expert, a regular domain user, or a programmer that developing software systems for a specific domain. Each of these users has a different background profile and a different role in the problem solution. Both are expected to impact the way these users use a DSL. We need comparable validation procedures to assess user experience with DSLs, in contrast with whatever was the previous problem solving approach in that particular context.

Comparing the impact of different languages in the software development process has some tradition in the context of GPLs (*e.g.*, [10]). Typically, the popularity of a language is used as a surrogate for its usability, but this simplistic approach is not particularly interesting for DSLs, which often have a well-bounded set of target users (*e.g.* people working in a particular organization) Another shortcoming of the "popularity" approach is that it does not help identifying the strengths and weaknesses of a language, be it DSL or GPL. Other sorts of evaluations on GPLs include benchmarks, feature-based comparisons and heuristic-based evaluations [10],[11]. Since the end users of GPLs are usually closer to computation concepts, while the end users of DSLs are generally closer to domain concepts of the context of use, these methods cannot be directly applied for DSLs either.

When usability problems are identified too late in the language development process, a common approach to mitigate them is to build tool support that minimizes their effect on users' productivity [12], [13]. Better Usability is a competitive advantage, although evaluating it remains challenging, because it is hard to interpret existing metrics in a fair and unbiased way.

When compared to using GPLs, the increased productivity achieved by using DSLs is the one of the strongest claims of the DSL community[3],[4],[14]. The problem is that this claim is mostly based on anecdotal reports on improvements that lack external validity. Other reports, present maintainability and extensibility improvements brought by a combination of DSLs and Software Product Lines (SPLs) [15]. The usage of DSLs has been favorably compared to the usage of templates in code generation, with respect to flexibility, reliability and usability [16]. In a recent survey DSL users reported that they achieved noticeable improvements in terms of reliability, development costs, and time-to-market [6]. Comparisons can also be made among competing DSLs: for instance, [17] compares a visual DSL against the textual language for which it is a front-end.

DSLs define a way for human to communicate with machines. Therefore, DSL evaluation should not be much different from evaluating a regular UI. We can argue that any UI is a realization of a language, where a language is considered as a theoretical object (a.k.a. model) that describes the allowed terms and how to compose them into the sentences involved in a particular human-computer communication. Examples of UIs range from compilers to command-shell and graphical applications, and in each of those examples we can deduce the human-computer (H/C) language that is being used to perform that communication [29]. The general goal for HCI is that "*it should increase efficiency of humans performing their duties within a computation infrastructure, without extra organizational costs, inconveniences, dangers and dissatisfaction, as well as undesirable impacts on the environment during long periods of learning, or maintenance, among others"* [18].

Usually, there is a broad spectrum of issues to evaluate Software's Quality. Looking at the quality standards, and to the current Software Evaluation techniques we can fit them to the particular case of DSLs. In the literature, most of the requirements are actually associated with a qualitative software characteristic called Usability. The need for development of Usability definition is discussed in several articles such as [19], [20]. The standards ISO/IEC 9241-11 (2001), ISO/IEC 9126 (2001) and ISO IEC CD 25010.3 [19] provide several definitions. The ISO IEC 9241-11 (2001) standard defines Usability as the *"extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"*. ISO IEC 9126 (2001) gives us a quality model for achieving 'Goal Quality', *i.e.*, Quality in Use. ISO IEC CD 250100.3 estimated that model into complete Quality Model [21], where Usability is considered part of Quality in Use. In the context of DSL's evaluation [22], important notions such as Quality in Use, internal and external Quality were considered strongly dependent on the DSLs' intended context of use [27].

DSLs are built for a more confined context of use, and they capture one particular set of domain concepts. When we evaluate these languages, the population of users is smaller, and the external validity of the result is expected to be much higher than we would have for a UIs. In the context of potential language's optimization procedure, we expect to find more relevant and accurate interpretations for these results.

III. Research Objectives and Methodological Approach

Despite the advantages that DSLs might bring to Software Engineering (by mitigating the accidental complexity of software), in order to be widely adopted by Software Engineering professionals, we need to provide the means to assess their Quality in Use and success of implemented problem solution when compared to the other solutions. The alternative is to accept the risk of developing inappropriate DSLs that can decrease the domain developers' productivity or even increase maintenance costs.

We need a rigorous collaborative procedure in order to evaluate DSLs (both during and after their development), as well as evaluate their sentences (called instance models). For that it is necessary to:

a) Define the quality criteria to evaluate DSLs;

b) Integrate in an existing IDE support for development of DSLs with high Quality in Use; and

c) Define a methodological approach to support the evolution of a DSL's design based on user experience and infer its impact on quality improvement during its lifecycle (*e.g.* traceability of design decisions).

We propose to build a comprehensive methodology that involves Usability concern in all phases of existing DSLs' development process. We should research the most suitable means to provide both reliable DSL evaluation metrics and iterative suggestions during DSLs' development and evolution. This methodology will be based on user-centric techniques and cope with the DSL's evolution by assessing the impact of the changes in the DSL's design and implementation on user experience. In order to be able to build this methodology it is necessary to answer the following questions:

- What are the relevant quality concerns for DSL's evaluation, and associated metrics? How can we take advantage of these metrics to actually measure the quality in use of a DSL? Which existing standard DSLs can we take as a reference for performing DSLs comparison (or comparison of software languages in general)?

- How to plan an effective experimental evaluation of a DSL (*i.e.*, giving statistically significant results with the minimum effort)?

- How to guide the software language engineer in order to build a DSL with high level of Quality in Use? What are the good language design patterns? How can we foresee the Usability of a DSL while in an iterative evolution step?

The methodology will be validated by compilations based on recommendations that emerge from it in the development of the DSLs and experimental assessment of their impact trough few case studies on the different DSLs.

We foresee the following main research activities that need to be applied in each development step of DSL in order to introduce Usability evaluation into development process:

A. Domain Analysis

The Domain analysis phase is needed in order to understand the domain in consideration, by collecting information about it. The output of these phase is a domain model [23], that represents the common and varying properties of systems within the domain, the vocabulary used in the domain and defines concepts, ideas and phenomena, within the system. Existing systems, their artifacts (such as design documents, requirement documents and user manuals), standards, and customers are all potential sources of domain analysis input.

In this activity, we find it essential to define and model DSLs target users and intended context of use. Also, we propose new models, *e.g.* scenario-based modeling and goal-oriented modeling, which are based on assessment of users' previous experience. They should be included into the existing domain analysis models in order to define the usability requirements and crucial tasks that should be supported by the DSL under evaluation. Also, we find it crucial to relate these requirements to dependent user and context models. These models should be considered from the beginning of the DSL's development process as quality criteria for the newly designed language. During the development process these models should be refined according to results of validation recommendations.

B. Language Design

Designing DSLs remains a difficult and under-explored problem [31]. Recent work has focused mainly on the implementation of DSLs and supporting tools. Also, Volter presents a collection of design patterns for describing the process of MDD. However, there still lacks detail for language design, development and implementation. We expect to contribute here with design patterns of Usability evaluation of DSLs.

In the Language Design activity, we propose to perform corpus evaluation of DSLs. Here, the main objective is to identify the means to evaluate the internal quality of a language, *i.e.*, in the perspective of language's evolution and validation. We expect to trace the impact of metamodel design changes, and collected statistic on the DSLs Usability.

C. *Testing – Controlled experiment*

The main objective of the testing activity is to identify the means to evaluate the Quality in Use of a language according to the requirement models described in the domain analysis phase. This involves the definition of experimental procedures/processes, heuristics and questionnaires. In order to be able to provide proper instrumentation for experimental evaluation, it is necessary to design support that will log Quality indicators, and present quantitative metrics result, so that developer is able to reason about the Quality in Use of implemented solution.

Designed instrumental support should be integrated into experimental model, so it can be validated trough controlled experiments. The quality in use of a language may be evaluated distinctly according to either its abstract syntax or concrete syntax which also implies the adoption of a (arguably) good interaction model. However, that is another aspect of usability evaluation of DSLs that is not part of this work. In scope of this work we find it necessary to evaluate only functional quality of concrete syntax, and not concentrate on evaluating concrete syntax by itself. Also, we will distinguish between evaluating a DSL from evaluating its implementing tool.

D. *Deployment and Maintenance - Collect and evaluate the Quality the Instance Models (sentences)*

The objective of this activity is to identify the appropriate means to qualify the instance models based on the users' feedback in the production environment. To be able to compare the (semantically equivalent) instance models expressed on the same language in a cognitive perspective we should revisit and improve corpus evaluation tools/techniques from testing activity. Also we should monitor the language's ability to support the evolution of the instance models without having negative impact of the languages usability.

E. *Validation - Iterative life-cycle*

The main objective of this activity is to build a conceptual framework to reason about the pertinence of the results of the language's Quality in Use in the overall language's life-cycle. It is important to identify what quality attributes (and corresponding metrics) have the most relevant impact on overall Quality in Use. We should evaluate impact of those quality metrics during following the language development step, as well as to validate suggestions for further improvements on the following steps. The framework should enable us to trace the impact of design changes on user experience with language and be interactively connected to the usability models proposed for another development activity.

By using existing language evaluation case studies we can compare the decisions from the reasoning framework, with the conclusions (considered sound by the community) taken from other language evaluation approaches. The expected output is a report containing a proof of correctness (completeness and soundness) of the conclusions taken by the reasoning framework on the observed case studies.

IV. Past Work and Preliminary Results

There are already many publications about UI Usability evaluation. However, we find that the Usability evaluation of a UI is typically superficial when compared to the required usability evaluation of DSLs. Existing methodologies do not cover all the relevant aspects and dimensions of usability evaluation, *e.g.* learnability, efficiency, effectiveness for all intended users and features of product. As it is hard to capture all the intended contexts of use for UIs at once, supporting tools are developed to support some parts of methodologies, usually built to provide questionnaires or collect some quantitative data, and are in most cases too general. Existing practices have very a low level of external validity, and sometimes it is hard to interpret what the collected information means, probably because of the wide spectrum of contexts of use that they target.

DSLs can have a precise definition of the end user's profile and task models, as well as syntactic models, that our method uses in order to achieve better results from its Usability evaluation. Moreover, we can rely on these results in order to validate the claim that DSLs can effectively narrow the gap between humans and computers, when compared to regular GPLs.

A. *Iterative user-centered design*

According to Mernik *et al.*, the Language life cycle consists of a set of phases [5]: *Decision*, *Domain Analysis*, *Design*, and *Implementation*. Visser adds *Deployment* and *Maintenance* to this process [23]. Besides adding *Testing* (as in any typical Software Product), we propose to introduce *Language Evaluation* just before *Deployment* [24]. This *Language Evaluation* phase is done with language quality concerns in an incremental and iterative user-centric approach, with the DSL end users, while crosscutting all of the involved phases, as suggested in [25].

By allowing significant changes to correct deficiencies along the development process, instead of just evaluating the DSL at the end of the process, when it might be too late, user-centered design can reduce the cost of development and support. The critical activities required to implement user-centered design are described in ISO 13407 [20]. Once the system is released to the users, an user experience assessment of DSLs and associated IDE may be highly beneficial [19].

An *Iterative Usability evaluation approach* should be merged with the DSL development cycle, as described in [22]. This approach supports reasoning about the already implemented and wished problem domain concepts of DSLs users. In a first moment, by defining them for the user and context models in the domain engineering phase, designing and implementing them in the language. In a later stage, the language concepts should be validated in the testing phase, along with the development environment proposed for using the DSL. Note that the combination of language and tool support is essential in the evaluation, because language usage will be significantly impacted by its tool support. As such, it is essential that the iterative usability evaluation covers both.

B. *Context-dependent evaluation*

Empirical evaluation with users, is recommended at all stages of development, or at least in the final stage of development [26]. To do so, we can use several methods, with different kinds of measures, where each type of measure is usually regarded as a separate factor with a relative importance that depends on the DSL's context of use [27]. These evaluations can be designed to target specific profiles of DSL users in order to increase their replicability.

For several predefined groups of DSL users we should use techniques like questionnaires, and observations to analyze the tasks involved while using a given DSL. Observations should include capturing quantitative indicators related to users' interaction with the DSL environment (*e.g.* mouse movements, keystrokes, heartbeats, or eye tracking). Experimenters in human factors have developed a list of tasks that can capture these particular aspects [28]. These tasks should be designed to capture relevant Usability concerns, *e.g.*, *effectiveness*, *efficiency* or *satisfaction*. We propose a systematic approach based on UIs experimental validation techniques to assess the impact of the introduction of DSLs on the productivity of its end users. To illustrate this evaluation approach we have presented a case study of a DSL for High Energy Physics [29].

C. *Experimental Language Evaluation*

We argue that the Quality in Use of a DSL should be assessed experimentally. In Software Engineering, a controlled experiment can be defined as "*a randomized experiment or quasi-experiment in which individuals or teams (the experimental units) conduct one or more Software Engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages or tools (the treatments)*" [30]. In the case of DSLs, this can be instantiated in early phases of development with domain experts that typically have to conduct with software construction, or evolution tasks. For the sake of comparing different languages, including the DSL under evaluation and any existing baseline alternatives to that DSL, representative user groups should be modeled and involved.

We proposed a general experimental evaluation model, tailored for DSLs' experimental evaluation, and its instantiation with several DSL evaluation examples [24]. These instantiations served as a proof of concept for the proposed experimental evaluation process. Our evaluation model can be instantiated for repeated evaluations of a DSL, thus building up a longitudinal evaluation of the DSL, while it evolves. This enables us to track and control the impact and scope of changes in the DSLs. The model also facilitates reasoning about which Usability levels are achieved for each user profile population, which can help language engineers in determining when the desired quality in use level is achieved (*i.e.* when additional changes do not have any more significant impact in the Usability of DSL). The representation of the evaluation as an instance of our evaluation model also facilitates the comparison of alternative DSL solutions, as well as the replication of previous evaluations, their approaches and decision models.

V. FUTURE WORK AND EXPECTED RESULTS

Our research will follow by proposing metrics and methodologies for Usability evaluation of DSLs, whose validity should be supported by real life experiments with users of existing DSLs. In order to do that, we find it necessary to define conceptual distance as the distance between concepts in the users' mind and the conceptual domain of a language. If we are able to measure that distance, and have methods that will minimize it, we can support the claim that DSLs are able to close the gap between domain experts and solution domain.

An additional step is to conceptualize models for performing DSL's evaluation *i.e.* quality model, instruments model, metrics and traceability model of design changes and their impact. This support should be tailored to internal and external quality attributes (such as syntactic and semantic models of the DSL under evaluation) and user's experience while using a DSL along several iterative evolution steps.

By providing that kind of support, we can effectively perform evaluation, whose outcome can be used to help increasing users' productivity, and explicitly model all the process. This evaluation procedure will give us faster convergence of language development, as we are able to monitor the impact of language evolution in the efficiency and effectiveness of practitioners using the language (and its companion toolset). As a side effect, we expect our evaluation work to contribute to the validation of the claim that DSLs are more usable then GPLs.

The impact of an evaluation process for DSLs is expected to be interesting from an industry point of view. With many organizations developing their own languages, or hiring companies to develop such languages for them, this framework will aid them in reaching more usable languages.

VI. CONCLUSION

Building DSLs is becoming very popular and by that there are increasing needs of some pointers in topic of their cognitive congeniality to end user. Although pragmatic, reactive approaches would not be necessary if domain experts could develop applications easily. It is necessary to explore more proactive approaches to improving DSLs'

Usability. We need to build a comprehensive methodology that support all phases of the Usability evaluation process and indicate ways to provide reliable metrics for supporting this evaluation. This is expected to enhance the community's awareness and recognition of the relevance of this topic in the process of SLE.

ACKNOWLEDGMENT

The authors would like to acknowledge CITI - PEst - OE/EEI/UI0527/2011, Centro de Informática e Tecnologias da Informação (CITI/FCT/UNL) - 2011-2012) - for the financial support for this work.

REFERENCES

[1] F. P. Brooks, "The Mythical Man-Month: Essays on Software Engineering", Addison-Wesley Publishing Company, 3rd Edition, 1995.

[2] M. Volter and T. Stahl, "Model-Driven Software Development". Glasgow, UK: Wiley, ISBN: 0-470-02570-0, 2006.

[3] S. Kelly and J.-P. Tolvanen, "Visual domain-specific modelling: benefits and experiences of using metaCASE tools," Proc. International Workshop on Model Engineering, at ECOOP'2000, 2000.

[4] D. M. Weiss and C. T. R. Lai, "Software Product-Line Engineering: A Family-Based Software Development Process". Addison Wesley Longman, Inc., ISBN: 0-201-69438-7, 1999.

[5] M. Mernik, J. Heering, and A. M. Sloane, "When and How to Develop Domain-Specific Languages," ACM Computing Surveys, vol. 37, No. 4, pp. 316-344, December, 2005.

[6] F. Hermans, M. Pinzger, and A. V. Deursen, "Domain-Specific Languages in Practice: A User Study on the Success Factors," Proc. 12th International Conference on Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science, Denver, Colorado, USA, October, 2009, pp. 423-437, doi: 10.1007/978-3-642-04425-0.

[7] P. Gabriel, M. Goulão, and V. Amaral, "Do Software Languages Engineers Evaluate their Languages?," Proc. XIII Congreso Iberoamericano en "Software Engineering" (CIbSE'2010), ISBN: 978-9978-325-10-0, Universidad del Azuay, Cuenca, Ecuador, April, 2010, pp. 149-162.

[8] J. Nielsen, and S. Gilutz, "Usability Return on Investment", Nielsen Norman Group, 4th edn. 2003.

[9] A. Marcus, "The ROI of Usability", in Bias, and Mayhew (Eds.): "Cost-Justifying Usability", North- Holland, Elsevier, 2004.

[10] L. Prechelt, "An Empirical Comparison of Seven Programming Languages," IEEE Computer, vol. 33, No. 10, pp. 23-29, October, 2000, doi: 10.1109/2.876288.

[11] D. L. Moody, "The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering", IEEE Transactions on Software Engineering, 2009, pp. 756-779.

[12] K. Y. Phang, J. S. Foster, M. Hicks, and V. Sazawal, "Triaging Checklists: a Substitute for a PhD in Static Analysis". Proc. Evaluation and Usability of Programming Languages and Tools (PLATEAU 2009), 2009.

[13] R. Bellamy, B. John, J. Richards, and J. Thomas, "Using CogTool to model programming tasks". Proc. Evaluation and Usability of Programming Languages and Tools (PLATEAU 2010), 2010.

[14] MetaCase: "EADS Case Study", http://www.metacase.com/papers/MetaEditinEADS.pdf, 2007.

[15] D. Batory, C. Johnson, B. MacDonald, and D. v. Heeder, "Achieving extensibility through product-lines and domain-specific languages: a case study," ACM Transactions on Software Engineering and Methodology, vol. 11, No. 2, pp. 191-214, April, 2002, doi: http://doi.acm.org/10.1145/505145.505147.

[16] R. B. Kieburtz, L. McKinney, J. M. Bell, J. Hook, A. Kotov, J. Lewis, D. P. Oliva, T. Sheard, I. Smith, and L. Walton, "A Software Engineering Experiment in Software Component Generation," Proc. 18th International Conference on Software Engineering (ICSE'1996), IEEE Computer Society, Berlin, Germany, March, 1996, pp. 542-552, doi: 10.1109/ICSE.1996.493448

[17] N. S. Murray, N. W. Paton, C. A. Goble, , and J. Bryce, "Kaleidoquery--a flow-based visual language and its evaluation", Journal of Visual Languages & Computing, 2000, 11, (2), pp. 151-189.

[18] T. Catarci, "What happened when database researchers met usability", Information Systems, 2000, 25, (3), pp. 177-212

[19] H. Petrie, N. Bevan, "The evaluation of accessibility, usability and user experience", in C. Stephanidis, (Ed.): "The Universal Access Handbook", CRC Press, 2009.

[20] N. Bevan, "Cost benefits framework and case studies", in "Cost-Justifying Usability: An Update for the Internet Age". Morgan Kaufmann, 2005

[21] N. Bevan, "Extending quality in use to provide a framework for usability measurement", Human Centered Design, 2009, pp. 13-22

[22] A. Barišić, V. Amaral, M. Goulão, and B. Barroca, "How to reach a usable DSL? Moving toward a Systematic Evaluation", Electronic Communications of the EASST (MPM), 2011

[23] E. Visser,: "WebDSL: A Case Study in Domain-Specific Language Engineering", in Book WebDSL: A Case Study in Domain-Specific Language Engineering' (Springer, 2007, edn.), pp. 291-373

[24] A. Barišić, V. Amaral, M. Goulão, and B. Barroca, "Evaluating the Usability of Domain-Specific Languages", in M. Mernik, (Ed.): "Formal and Practical Aspects of Domain-Specific Languages: Recent Developments", IGI Global, 2012, in press.

[25] C. Atkinson, and T. Kühne, "Model-Driven Development: A Metamodeling Foundation', IEEE Softw., 2003, 20, pp. 36-41

[26] J. Nielsen, and R. Molich, "Heuristic evaluation of user interfaces". Proc. SIGCHI Conference on Human Factors in Computing Systems: Empowering People (CHI'90), Seattle, Washington, United States, 1990.

[27] A. Barišić, V. Amaral, M. Goulão, and B. Barroca, "Quality in Use of DSLs: Current Evaluation Methods". Proc. 3rd INForum - Simpósio de Informática (INForum2011), Coimbra, Portugal, September 2011.

[28] P. Reisner, "Query languages": in "Handbook of Human-Computer Interaction", North-Holland, 1988, pp. 257-280.

[29] A. Barišić, V. Amaral, M. Goulão, and B. Barroca, "Quality in Use of Domain Specific Languages: a Case Study". Proc. 3rd ACM SIGPLAN workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2011), Portland, USA, October 2011 pp. 65-72

[30] 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, No. 9, pp. 733-753, September, 2005.

[31] M. Pfeiffer, and J. Pichler, "A comparison of tool support for textual domain-specific languages", UAP Printing Solutions, 2008, pp. 1-7.

[32] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley Publishing Company, ISBN, 1995.

2012 Eighth International Conference on the Quality of Information and Communications Technology

Object-Functional Patterns: Re-Thinking Development in a Post-Functional World

Tiago Boldt Sousa
Department of Informatics Engineering
University of Porto
Faculty of Engineering
INESC TEC (formerly INESC Porto)
Rua Dr. Roberto Frias, s/n
4200-465 Porto Portugal
tiago.boldt@fe.up.pt

Hugo Sereno Ferreira
Department of Informatics Engineering
University of Porto
Faculty of Engineering
INESC TEC (formerly INESC Porto)
Rua Dr. Roberto Frias, s/n
4200-465 Porto Portugal
hugo.sereno@fe.up.pt

***Abstract*—Programing paradigms define how to think and design while creating software. Object-Oriented and Functional paradigms are two of the most adopted for synthesizing it. Modern languages, attempting to provide higher abstractions, are increasingly supporting native multi-paradigm programming styles. The Object-functional approach still uses classes for information and high-level structure, but allows algorithms to be implemented functionally. New challenges now exist and there is a general lack of knowledge on best practices for adopting this paradigm. This research proposes the systematic usage of software patterns to capture these new recurring problems and their solutions, though not discarding the identification of new algorithms and designs. We will use Scala as a base language, and will attempt to validate our hypothesis through multiple methodologies, including quasi-experiments and case studies. We expect to provide a basis for improvement for programming languages (through pattern absorption) and for software engineering professionals.**

***Index Terms*—Software Engineering, Programing Paradigms, Design Patterns**

I. Introduction

Synthesizing software is a problem that can be approached through multiple paradigms. Object-Oriented and Functional are probably the two most adopted currently. Providing different thinking styles, they have previously been mutually exclusive. Nevertheless, these two paradigms are not incompatible with each other and modern languages are now providing support for them to be used together in a multi-paradigm approach, referred to as Object-Functional. This new paradigm provides advantages from both sides, allowing the high-level structure to be modeled using Classes, while still exploiting a functional algorithm definition. Adopting such paradigm opens new research possibilities regarding software engineering best practices for documenting best practices while combining both paradigms, avoiding probable mistakes such as users biased from previous experiences using either of the paradigms. We believe that such combination can be better than the sum of its parts, when correctly combined. Software design patterns are a generally accepted way to share software engineering knowledge and could be used to document recurring problems using the object-functional paradigm. Furthermore, we believe that known patterns, such as the ones introduced by Gamma et al [1] in the book *"Design Patterns"* can be tampered with, with slight adjustments in the forces, resulting in additional solutions that benefit from the multi-paradigm approach.

This work proposes to evaluate the possibility of evolving known software patterns, as well as identifying and documenting new ones, in order to adapt them to the object-functional paradigm. It is the authors belief that such approach might positively influence the work quality for developers, with a multitude of measurable advantages through software engineering metrics, such as: increased production efficiency, reduced code size, and the production of less error-prone applications. A reference implementation of such patterns will be provided using the object-functional language Scala.

This document is organized in seven sections. After the introduction, section II describes the motivation for this proposal. In section III a small introduction to the topics researched is presented, followed by section IV describing the current state of the art. Our thesis proposal is better described in section V. Sections VI and VII present the past and future work, respectively. The document finishes with its conclusions in section VIII, where final thoughts about this work are presented.

II. Motivation

Object-functional languages provide a bridge between two highly adopted programming paradigms: Functional and Object-Oriented. We believe that this recent paradigm can improve the quality of code generated by programmers by avoiding state and mutable data as functional programming does, while still providing programmers with the intuitive OO Class-oriented way representing data.

Despite the advantages introduced by object-functional languages facing its separated composing paradigms, little work has been found attempting to document patterns in this context. Regarding such, we can raise the following questions: How could known software patterns be implemented in a object-functional language? How would these be improved, while compared to their more generic, plain OO-based description? What new patterns could emerge in this context?

This proposal describes an attempt to find answers for such questions, attempting to provide a positive contribution to the research area of software engineering. For that, we will aim at providing novel patterns to be applied in this context, which can be of use for researchers and professionals working with these technologies, as well as provide patterns feasible for being absorbed by programming languages themselves in the future, increasing their abstraction level, hence, simplifying programmers' work.

III. Background

A. Programming Paradigms

A programming paradigm defines the thinking style applied while programing. Different paradigms offer different concepts and abstraction to represent elements within the program (such as variables, functions or objects). Programming languages can adopt simultaneously multiple paradigms, providing developers with an increased freedom to use the style it best fits them, or the problem in hand.

1) Object-Oriented Paradigm: The object oriented paradigm was first introduced as part of Ivan Sutherland's PhD thesis [2] but formalized only later in the first version of the Simula language in 1967 [3] . Soon it was implemented by Alan Kay in the Smalltalk, a fully Object-Oriented language. This paradigm allows programmers to model data as classes, providing an intuitive way to model information as it is observable in the real world, through the notion of objects that have properties and perform actions. OO promotes code reuse as classes are easily portable between projects that need to model the same information. Mitchell [4] describes OO as a set of four key features: dynamic dispatch, abstraction, subtype polymorphism, and inheritance.

2) Functional Paradigm: The Functional paradigm was introduced by John McCarthy in the 50's through the Lisp programming language [5] . Functional programming languages model applications with a mathematical stance, promoting equational reasoning, making them easier to formally proof. Contrary to other computation models, this paradigm avoids keeping a state or mutable data in the program, with every computation being made only regarding the inputs provided to a function and logic being handled as a composition of functions. It is common for Functional programming languages to have advanced type and type-inference systems, such as Hindley-Milner [6] , which not only considerably reduce the amount of code needed, but also provide a stronger validation at compile time when compared to other non strongly-typed languages. Performance improvements are also relevant, with lazy evaluation being key, computing data only when it is needed by other computations, hence, providing the ability to handle concepts such as infinite data streams, as well as parallelization being freely achieved through the use of the multiple provided parallel data structures.

3) Object-Functional Paradigm: Languages adopting the object-functional paradigm are actually multi-paradigm languages that merge the best of the object-oriented and functional paradigms, providing developers with the ability to represent data using the classes provided by OO and implement their algorithms using the more mathematically approach of functional programing, retaining the features above mentioned. The improved type systems also reduce the proneness to error. Several languages have been adopting both these paradigms, with Scala being one of the most actives.

B. The Scala Programming Language

1) Overview: Scala is a multi-paradigm, general purpose programming language, designed to express common programming patterns in a concise and type-safe way. By joining the functional and object-oriented paradigm, Scala could enable programmers to be more productive at their work when correctly applying both paradigms together.

2) The Expression Problem and Scala: Originally described by Wadler [7] , the expression problem is well explained by Torgersen who formulates it as: "Can your application be structured in such a way that both the data model and the set of virtual operations over it can be extended without the need to modify existing code, without the need for code repetition and without runtime type errors" [8] . This is a recurring problem in single paradigm languages, specifically, pure object-oriented and functional languages. The expressiveness of a programming language is a relevant factor for guaranteeing code maintainability. By 2005, Nielsen et al [9] , evaluated the language's expressiveness; they considered Scala to be able to solve the expression problem, a positive influence regarding our programming language choice.

C. Software Patterns

The concept behind Software Patterns was invented by Christopher Alexander in the civil architecture domain. Alexander stated that *"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice"* [10] . Software patterns follow the same principle, with a pattern being a detailed description of a problem, its context, variating forces and a proposed solution. Meszaros [11] describes the components for patterns in detail as follows:

1) Context: You are an experienced practitioner in your field. You have noticed that you keep using a certain solution to a commonly occurring problem. You would like to share your experience with others.

2) Problem: How do you share a recurring solution to a problem with others so that it may be reused?

3) Forces:

- Keeping the solution to yourself doesn't require any effort. Sharing the solution verbally helps a few others but won't make a big impact in your field.
- Writing down your understanding of the solution is hard work and requires much reflection on how you solve the problem.
- Transforming your specific solution into a more widely applicable solution is difficult.
- People are unlikely to use a solution if you don't explain the reasons for using it.
- Writing down the solution may compromise your competitive advantage (either personal or corporate.)

4) Solution: Write down the solution using the pattern form. Capture both the problem and the solution, as well as the reasons why the solution is applicable. Ensure that the necessary information is communicated clearly and include optional elements when needed to capture any additional useful information. Distribute the resulting pattern to the largest audience you feel it could help that does not compromise your competitive advantage. Often, this means publishing your patterns exclusively within your company via Intranets or company journals.

IV. State of the Art

A. Functional Patterns in Object-Oriented

Kuhne in 1999. stated in his PhD thesis that "Design patterns inspired by functional programming concepts can advance object-oriented design" [12] . With his thesis, Kuhne demonstrated the feasibility of porting some concepts from the functional paradigm to a purely object-oriented language, by actually implementing functional concepts as usable objects in a non functional language. In his pattern language, Kuhne describes and relates the following patterns, which are common functional concepts:

- Function Object;
- Lazy Object;
- Value Object;
- Transfold;
- Void Value;
- Translator.

Implementing those functional concepts inside a purely object-oriented language motivated the usage of a functional style even outside functional languages, which he proves to be feasible and advantageous for programmers.

B. Design Patterns in Scala

A more recent research was performed by Fredrik Løkke, with his masters thesis, where he implemented the patterns described in *Design Patterns* by the Gang of Four, in the object-functional language Scala [13] . With his work, Løkke exploited the functionalities of the language, such as lazy evaluations, generics, case-classes amongst others to implement the patterns in a more functional way. The author states that there were some patterns absorbed by the language itself, such as the Singleton pattern, showing how these might influence the evolution of programming languages. In his conclusion, the author highlights the relevance of the higher abstraction models that the language provides and suggests that these models can be a starting point for the creation of new patterns that can facilitate the creation of powerful software at lower costs.

C. Functional Patterns in Scala

In the book "Scala in Depth" [14] , Suereth presents a review on functional patterns using the (object-functional) Scala programming language. Described are functors, monads and applicative functors. Such patterns could pose as starting ground for research on object-functional patterns, either by observing natural evolution of these patterns in the wild or by identifying other patterns that depend on the ones presented in the book.

D. Design Patterns as Higher-Order Datatype-Generic Programs

Design pattern solution are usually extra-textually presented, through prose, pictures or prototypes. Gibbons [15] believes that this happens due to the lack of expressiveness from programming languages that, otherwise, could be used by themselves to describe solutions in patterns, providing directly reusable library components, presenting multiple benefits: patterns may be reasoned about, type-checked implementation, applied and reused, just as any other abstractions can. Higher-order parameterization and datatype-generics would provide the customization needed to adapt the provided patterns implementation to the specific problem at hand. Considering this, Gibbons presents a functional implementation in Haskell of four highly-related GoF patterns (Composite, Iterator, Visitor and Builder), promoting the uptake of higher-order and datatype-generic techniques, encouraging their incorporation in mainstream programming languages.

V. Thesis Proposal

A. Problem Description

Considering the lack of documented knowledge regarding the Object-functional paradigm, we believe that software patterns, as an accepted approach to share knowledge and propose solutions for solving common software engineering problems, should be used to collect and disseminate knowledge on the subject. Considering this, the following questions can then be raised:

- Is it possible to improve known software patterns following the object-functional paradigm?
- What knowledge can we provide to promote more reliable and less error prone implementations?
- What new patterns can emerge in this context?
- How would Object-functional be an improvement over the traditional purely object-oriented or functional implementations in similar contexts?

We believe that documenting best practices for using the object-functional paradigm could result in less error-prone and more efficient implementations of software artifacts. Moreover, current research let us believe to be possible to document several new patterns regarding this paradigm, improving both in algorithm programming style and data organization imposed by the purely Object-Oriented or functional formalizations.

B. Hypothesis

Assuming (i) that we want to increase the efficiency while synthesizing software, (ii) that patterns are a proven way of capturing empirical knowledge on best-practices, and (iii) that a pattern language empowers a more abstract, general, and hence powerful way of reasoning, then: *If programmers are provided a pattern language for implementing software exploiting the object-functional paradigm, when compared to traditional (either strict OO or Functional) approaches, they will (i) produce software more efficiently, (ii) produce less error-prone artifacts, and (iii) object-functional expressiveness of these patterns will promote higher quality regarding software engineering metrics.*

C. Research Objectives

This research will focus on researching Software Patterns, both existing and new ones, using Object-Oriented and Functional programming languages.

1) Object Functional Programming in program synthesis: Understand how the OFP paradigm changes program synthesis by comparing software engineering metrics such as code size, execution time, anti-patterns introduced, amongst others;

2) Improve existing patterns: Research if and how existing patterns could be reformulated to take advantage from the OF paradigm;

3) New object-functional patterns: Identify and document new patterns or pattern languages oriented in this context;

4) Reference Implementation: Provide a reference implementation of the patterns and pattern language(s) described;

5) Reproducible description of the deliverables: Provide reproducible experimental packages for the attained deliverables, namely, the point previously described.

D. Methodological Approach

The validation of this work will be attempted using two different methodologies: case studies and quasi-experiments.

1) Case Studies: The first form of validation for this work will consist of case studies performed on industrial partners. We expect to observe an improvement in their productivity and overall code quality considering multiple generic software engineering metrics as a result of the adoption of the object-functional paradigm, the implemented library and application of patterns documented in this thesis. We will use data gathered from other projects from the same team where object-functional languages are not used to evaluate the performance increase with the new paradigm.

2) Quasi-Experiments: In order to measure the ability from developers to adopt the object-functional paradigm and interpret the documented patterns, quasi-experiments will be held in a controlled academic environment. We intend to create homogeneous groups of students and provide them with a set of problems and corresponding documented patterns to be solved with one of the three paradigms: object-oriented, functional or object-functional. Implementations will then be evaluated regarding efficiency, effectiveness, anti-patterns produced, amongst other metrics. This data will provide us with the possibility to evaluate if and in what way is adopting the object-functional paradigm effective amongst non-biased developers.

E. Scientific Impact

Scientific contribution from this research can be published in conferences and journals on general software engineering or specific conferences on programming paradigms or patterns. Appendix A lists some of the conferences and journals where our research is feasible to be published. We believe that the publication of new patterns, pattern language(s), their reference implementation and our results can greatly contribute to both academic researchers working in the same subject or professionals that might find use in the patterns documented.

VI. Past Work and Preliminary Results

This project is still in its early stage. Currently, research is being conducted regarding programming languages, patterns and pattern languages in general, exploring their current state of the art. Knowledge gathered from this research has been used to contribute to the European project eCAALYX. In this context, a paper was accepted in the Cooperative Design, Visualization and Engineering conference entitled "Scalable Integration of Multiple Health Sensor Data for Observing Medical Patterns". The paper describes the concept of a Timeline data structure that strongly exploits the object-functional paradigm, implemented with the Scala programming language.

VII. Future Work and Expected Results

A. Patterns and Pattern Language

With this thesis, we intend to document multiple object-functional patterns and a pattern language relating them. We expect these patterns to be of use for researchers working under the same area as starting point for their own research and to professionals as a source of knowledge for helping them solve the documented recurring engineering problems.

B. Reference Implementation

For all patterns documented, a reference implementation will be produced. This will exemplify the usage of the pattern, as well as provide the artifact used during this research.

C. Verification and Validation

Verification and validation of this research, as stated in section V-D, will be achieved through case studies and quasi-experiments. Results from both validation methodologies will allow us to evaluate how the paradigm under study benefits, or not, the development of software artifacts. We expect to observe the following results:

- Reduction of code size, achieved due to the higher abstractions provided by the paradigm;
- Reduction of Anti-Patterns and bugs introduced, through the strongly-typed system;
- Reduction of development time, enabled by the usage of the reusable pattern library provided.

VIII. Conclusions

Patterns and pattern languages have long been accepted by software engineers as a useful source of information to solve software problems. Still, modern languages are providing developers with new paradigms or combinations of paradigms that allow them to do their work better, with less effort. This research will focus on the object-functional paradigm (particularly using the Scala language) and will document best practices while joining the two well known paradigms (OO and Functional) as patterns. We expect to either improve existing patterns in this context, as well as identify new ones, providing a source of knowledge for developers and researchers working in the area. We expect to validate this thesis through the application of case studies with industrial partners and quasi-experiments in academic environments.

References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*, ser. Addison Wesley Professional Computing Series, A.-W. P. Co, Ed. Addison Wesley, 1995, vol. 47, no. February. [Online]. Available: http://www.amazon.co.uk/exec/obidos/ASIN/0201633612/citeulike-21

[2] I. E. Sutherland, "Sketch pad a man-machine graphical communication system," in *Proceedings of the SHARE design automation* ..., 1964. [Online]. Available: http://dl.acm.org/citation.cfm?id=810742papers2://publication/uuid/C4EB193D-8F79-419A-B6E6-143A0D547CD8

[3] J. R. Holmevik, "Compiling SIMULA: A Historical Study of Technological Genesis," *Ieee Annals Of The History Of Computing*, vol. 16, no. 4, pp. 25–37, 1994. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=329756

[4] J. C. Mitchell, *Concepts in Programming Languages*. Cambridge University Press, 2003, vol. 45, no. 2007. [Online]. Available: http://www.holtsoft.com/books/java_concepts.html

[5] J. McCarthy, "History of LISP," *Sigplan Notices*, vol. 13, no. 8, pp. 173–185, 1978. [Online]. Available: http://portal.acm.org/citation.cfm?id=1198360

[6] L. Damas, "Type Assignment in Programming Languages," Ph.D. dissertation, University of Edinburgh, 1985.

[7] P. Wadler, "The Expression Problem," pp. 1–4, 1998. [Online]. Available: http://homepages.inf.ed.ac.uk/wadler/papers/expression/expression.txt

[8] M. Torgersen, "The Expression Problem Revisited," *ECOOP 2004–ObjectOriented Programming*, vol. 3086, pp. 1–44, 2004. [Online]. Available: http://www.springerlink.com/index/H73P577R36J8QHMM.pdf

[9] K. A. Larsen, "Types in Object-Oriented Languages The Expression Problem in Scala," *Knowledge Creation Diffusion Utilization*, 2005.

[10] C. Alexander, *A pattern language*, ser. Center for Environmental Structure series ; 2, S. Ishikawa and M. Silverstein, Eds. Oxford University Press, 1977, vol. 21, no. 2. [Online]. Available: http://books.google.co.uk/books?id=hwAHmktpk5IC

[11] G. Meszaros, "A pattern language for pattern writing," *Pattern languages of program design*, 1998. [Online]. Available: http://xunitpatterns.com/~gerard/plopd3-pattern-writing-patterns-paper.pdfpapers2://publication/uuid/420266E9-5BD8-41A3-AA9B-F03763E9E78E

[12] T. Kuhne, "A Functional Pattern System for Object-Oriented Design," Ph.D. dissertation, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.1134&amp;rep=rep1&amp;type=pdf

[13] F. S. Lø kke, "Scala & Design Patterns," 2009.

[14] J. D. Suereth, *Scala in Depth*. Manning Publications, 2012.

[15] J. Gibbons, "Design patterns as higher-order datatype-generic programs," *Proceedings of the 2006 ACM SIGPLAN workshop on Generic programming WGP 06*, no. December, p. 1, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1159861.1159863

Appendix

A. Conferences and Journals

There are several conferences and journals to which our research is of interest:

1) Journals:

- Transactions on Software Engineering (IEEE)
- Transactions on Software Engineering and Methodology (ACM)
- International Journal of Software Engineering and Knowledge Engineering
- Journal of Systems and Software (Elsevier)

2) Conferences:

- Conference on Pattern Languages of Programs (PLOP) (many)
- Systems, Programming, Languages and Applications: Software for Humanity (SPLASH)
- Object Oriented Programming, Systems, Languages and Applications (OOPSLA)
- International Conference on Functional Programming (ICFP)
- European Conference on Object-Oriented Programming (ECOOP)

Ontologies for Product and Process Traceability at Manufacturing Organizations: A Software Requirements Approach

José C.C. Martins
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
joseccmartins@gmail.com

Ricardo J. Machado
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
rmac@dsi.uminho.pt

Abstract— **A Traceability business process is mandatory and unavoidable on manufacturing organizations. Customers, particularly original equipment manufacturers, require it on contracts, while governments enforce it, through rules and regulations.**

Organizations fail to create and sustain a business process satisfying traceability demands. IT departments are one of the main players on efforts to create a solution, as this process is only manageable when supported by software. This document presents an approach to improve the understanding of traceability business process by using ontologies as a requirements modeling technique.

Keywords- software requirements, computer science related discipline: information management, management related discipline: information systems

I. INTRODUCTION

Traceability on a manufacturing organization, aims the persistence of the relevant information related with the organization core activities. Nowadays a traceability business process (BP) is mandatory and unavoidable on any organization acting as a product provider. Externally, it is explicitly required on customer contracts, particularly when established with original equipment manufacturers (OEM). Governments, also, enforce it through rules and regulations. Internally, to pursue continuous improvement and answer the requirements of increased efficiency, it is necessary to track the manufacturing activities information with high accuracy and detail.

Organizations face several difficulties to implement and sustain a business process satisfying traceability demands. The roots of main difficulties lie on the lack of understanding and agreement by main players on the meaning of traceability concepts, concrete demands, and the process nature itself. Traceability is not a new concern, yet it cannot be considered well understood and defined.

The relevance of this research topic was already recognized on academic and business fields. European Commission's invested €100M on projects TRACE [1] and PETER [2], to increase research on food traceability. GS1, an international not-for-profit association composed by multinationals, retailers and manufacturers, created a Global Traceability Standard on 2006 [15].

IT departments are one of the main players to provide a solution, as this process only becomes manageable when supported by software applications. An organization possessing a degree of operational complexity that require software solutions to handle its manufacturing and logistics activities, cannot cope with traceability on a manual approach supported solely by paper work [45]. Besides, higher complexities on operational activities (e.g. raw material income, lot use, resources parameters) enforce the support of software solutions [46].

However, the development of these solutions is compromised, since requirements elicitation, by the lack of understanding on appropriate support to this process. Current research aims to facilitate the development of software solutions to support traceability BP, and along process to provide artifacts that act as enablers on organizational efforts to implement this business process. This PhD thesis addresses the knowledge improvement of traceability BP, supported by core artifacts, such as an ontology of traceability BP and respective taxonomy, which are expected to provide a common and improved understanding to all players, and become particularly valuable along the requirements elicitation efforts.

Traceability core activities are deeply connected with information handling: acquire, relate, persist and provide [35, 43]. Advances on these areas, through new artifacts, techniques or methods may positively feedback traceability process, triggering and sustaining its improvement [9].

This manuscript sustains that significant traceability problems are addressable through software engineering research, namely, the construction of domain models based on ontologies. Resulting outputs will benefit software engineering and business (e.g. Quality, Logistics, and Operations) fields simultaneously. Also promising is the potential to create new artifacts to software developers.

Next section provides a summary of relevant literature on traceability business process (traceability BP) in manufacturing organizations. Third section addresses the research objectives and the methodological approach. Section 4 briefly describes past work and preliminary results. In Section 5 future work and expected results are presented. Last Section depicts some conclusions.

II. State-of-the-Art

This section presents traceability BP state-of-the-art, the main problems a manufacturing organization encounters to set it up and how research addresses them so far.

A. *Definition and Goals*

On this document, traceability is understood as "the ability to track forward the movement through specified stage(s) of the extended supply chain and trace backward the history, application or location of that which is under consideration" [15].

The traceability responsibilities are the identification and trace of the history, distribution, location, and application of products. A traceability system must record and follow the trail as products that come from suppliers, are processed and distributed as end products.

Traditionally, its main purpose is linked to product recall: "... a procedure to withdraw all products with a particular deficiency from the supply chain" [24].

But traceability may serve many other organization processes. Töyrylä identified applications that benefit from traceability data (Table I.). These applications, consumers of traceability information present the broad range of its usage.

Traceability can protect a producer from product liability claims, providing the evidences necessary to prove law requirements were completely fulfilled. It may also serve to demonstrate a product origin or flow. Proof-of-origin usually aims to satisfy market demand for information [10, 12].

Traceability data can be the primer input to monitor, control and manage organizational quality and processes, also its information may become a solid support for their improvement [8, 24]. Proof-of-quality implies the ability to provide evidences on quality assessments realized on manufacturing process. An organization may reject the responsibility on failures based on these evidences. Besides, it may also self-promote a Quality image toward its customers, becoming quality certified [10, 24]. Traceability information can provide the basis to identify security breaches, through the products' monitoring along its supply chain, and enable the identification of counterfeit and illegal items. It may also be used to track moments or locations along the supply chain where products are prone to suffer damages or be deviated.

TABLE I. Traceability Consumer Applications

Consumer Applications	Guard	Promote
Recall	x	
Product -liability-prevention	x	
Quality- and process-improvement	x	x
Proof-of-quality and proof-of-origin	x	x
Logistics		x
Security	x	x
After-sales	x	
Accounting		x

On logistics, traceability information may be used to optimize material routes and improve planning and management, mainly due to improved links to the other organizations with whom there is collaboration.

Warranty data may be handled on Traceability, linked to a product, and serving as input to after-sales.

Traceability may work with accounting applications to evaluate inventory or with controlling applications to identify process inefficiencies.

Traceability information, on a manufacturing organization, protects or limits the damage and costs if a problem occur, menacing the organization [14]. Simultaneously it also sustains the organization change management process and respective improvement efforts as presented on table A1 [24, 28, 31].

There is another important traceability BP responsibility that does not specifically fit prior classification. This process must implement and obey government regulations, laws, customer requirements and standards (mandatory for respective certification) that directly address traceability.

B. *Relation with Software Solutions*

According to Töyrylä, "technical enablers include the computerization of data processing and the use of automatic identification in data collection." The need to ensure "Long-term availability of data" and "the frequency, quickness and accuracy of the information collection" address directly data persistence and recording responsibilities of software solutions [45].

Software solutions are also enforced by the need of fast response times, particularly when retrieving data [11]. On manufacturing environments, traceability activities must be synchronized with production infrastructure and respective operations [28]. The automation of manufacturing enforces a similar approach on related traceability activities. Panetto [38] suggested that any manufacturing software solution should have traceability data acquisition embedded.

Neto [35] and Terzi [43] stated that traceability activities are information management activities, rendering IT knowledge applicable on the study and improvement of traceability itself.

Buhr [9] recognizes it is not only the traceability process that pulls software solutions with supporting needs. The information technology revolution exemplified by the Internet and the underlying information-technology hardware (e.g., increased computer processor speeds, increased data-storage capacity, electronic data capture and measurement devices) push and enable traceability process to wide its scope and detail. Terzi [44] identified new technologies which applied on product identification leverage traceability software solutions to more detail and accuracy.

C. *Opportunities*

Despite their best efforts, manufacturing organizations face several vicissitudes when implementing a traceability business process. Some of the difficulties are related with the support of traceability BP by software solutions, and root problems specific of IT field. In parallel they also raise opportunities that are better tackled through software solutions. Main challenges, identified on literature, are presented hereafter.

Ideally "all information regarding products is recorded" [38]. However the size of traceability data has an impact on the respective required management effort. To improve traceability efficiency or even to render it practicable, the quantity and quality of the information that should be collected must be reduced to a manageable and appropriate amount. This evaluation must be sustained on sound knowledge of the traceability BP and the relative interest of subjects to trace [24, 31, 46].

A software supporting traceability must be able to receive, identify and handle data, regards its type [21, 26, 38, 43, 44, 46]. "The heterogeneity of applications managing information (ERP, PDM, MES... [1]), of users transforming, using and producing information (different operators), even of the meaning, the same information may address on different domains of pertinence (business or manufacturing), raises difficulties to the information recovery, leading traceability systems to fail at collecting information" [46]. Interoperability problems are outcomes of the differences between organizational units and between partner organizations.

Traceability is deeply interconnected with other business processes. The product/process data to trace is embedded in the activities included on other organizational processes [38, 41]. Due to the pervasiveness of traceability activities, respective responsibility is spread among several organizational unit, each one with different interests and approaches [44].

Traceability BP is not limited to a single organization boundary with a single set of traceability syntax, semantics, and concepts [16, 26, 27]. Also on the organizations network traceability requirements must be balanced with security, or secrets constrains [41].

As Gampl [14] states the organization' management lack a clear knowledge of the traceability nature. This lack is common also among stakeholders giving birth to vague, fragmented and incomplete requirements [8, 33].

Several efforts were developed to minimize the lack of knowledge problem. Various enterprises join together and defined a Traceability Business process standard [15]. ISO standards refer traceability and certify its implementation [5]. SAP summarized traceability best-practices [42], and European community issued new regulations [13]. All these documents contain valuable knowledge to guide the efforts to implement traceability.

III. Research Objectives and Methodological Approach

A. Research Objectives

The literature review revealed that lack of knowledge, on traceability BP, besides being a constraint on organization efforts towards its implementation, also was the root or acted as an amplifier of other perceived difficulties. Within the development of software solutions, the work of several players, is polluted by misunderstandings and fragments of traceability concepts. These difficulties have high impact on developers of software solutions, and interfere on the early stages of a solution development, on requirements engineering and design [19].

Traceability BP body of knowledge is currently scattered among several initiatives such as ISO standards, European regulations, and best-practices [1, 5, 13, 15, 42]. Regards the richness of contained knowledge, this documentation is hard to apprehend and use on the context of IS application domain. "There is a clear need to make them more abstract and to define methodologies in order to facilitate understanding of their defined concepts" [38].

The improvement of traceability understanding from the software developer's point of view will reduce lack of knowledge about what the system should do the technological options and the future situation [33]. It will also reduce "misunderstanding of concepts, ideas and definitions, making use - whenever possible - of shared standards" [44].

Tursi [46] propose the use of an "Ontology for the representation of domain's knowledge, in order to ensure a non ambiguous understanding of objects and concepts". A traceability BP ontology providing the domain concepts and relationships among them (conceptual relations) provides an adequate solution to address this difficulty [7].

Gasevic recognized that existing ontology development methodologies are fairly general and only suggest steps to be followed [17, pag.65]. Resulting ontologies tend to be very sensitive to their developers skills, and specificities of the environment where the knowledge is acquired. For the purpose of this research the resulting ontology must be general and independent of any particular organization characteristics. Thus the development process must be repeatable and result on similar traceability ontologies despite their developers or the environment where it occurs.

Therefore, the first research question is:

Q.1. How to create a traceability business process ontology?

Contained on this ontology, are general characteristics, recognizable as adequate properties of a software solution, aiming the support of traceability BP. Characteristics that refer the purpose, the needs, the goals, the functionalities, the constraints, the qualities, the behaviors, the services, the conditions, or the capabilities, and may ground a process to identify a software solution requirements [25].

The specific needs on the software to support traceability for a particular organization are only possible to obtain through requirements elicitation. From this effort, however, it is also common to collect overlapping or conflicting requirements, all together with others that are isolated and that do not make sense on domain.

This ontology, providing a more abstract and global overview of the domain, may be used to drive and focus the refinement of requirements, identifying the gaps where additional requirements should procured or even completing them. It may also clarify the conflicts between requirements. Main challenge is how to juxtaposing the detail, specific

[1] Enterprise Resource Planning (ERP),Product Data Management systems (PDM),Manufacturing Execution Systems (MES)

requirements from organization stakeholders with the ontology broad domain mapping.

Second research question is brought by this possibility:

Q.2. How to infer and validate the requirements and models of a Traceability software solution from respective ontology?

The identification of traceability BP requirements was also proposed on Terzi [44], Samarasinghe [41], and Khabbazi [28] studies. On a parallel approach, Ramesh [39, 40] proposed "a framework for a traceability based knowledge management system to support the design, customization and delivery of information product and e-service families".

B. *Methodological Approach*

The previous literature review identified, that problems addressed on current study, were already described and explained. However they are not yet solved. To reduce their significance, and simultaneously improve the understanding of traceability phenomena, an adequate strategy is to prescribe solutions to these particular problems and create artifacts that embody those prescriptions [34]. This strategy belongs to design science paradigm. It is focused on business needs and in utility. Also the goals aimed by presented research questions are appropriate to be pursued through Design Science, as it "seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" [18]. Hevner cleared that these "IT artifacts are broadly defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)" [18].

An important characteristic of Design Science is its pro-activeness with respect to technology, attempting to lead the evolution of software research and not merely react to it [18, 22].

Therefore, the proposed study will be structured according Design Science Research (DSR) methodology.

DSR uses an iterative approach (see Fig. 1) beginning with the Awareness of a Problem, a solution is created, drawn abductively from existing knowledge. The rigor of DSR is derived from the effective use of prior research (existing knowledge base) [18]. Solution and respective Artifacts are evaluated through metrics that instantiate the research goals [34]. These steps are repeated until a satisfactory solution to problem is found.

On research conclusion the knowledge acquired during process is consolidated, discovered through the detection and analysis of contradictions, only present on the specific act of constructing [30].

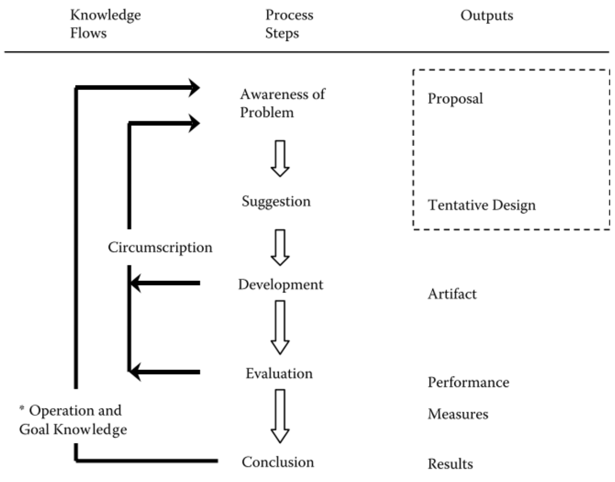

Figure 1. Design Science Research Cycle (Vaishnavi [30]).

IV. PAST WORK AND PRELIMINARY RESULTS

This research aims the development of new knowledge, in parallel with artifacts that uphold the development of IS solutions supporting traceability BP. It pursues Hevner's principle [18], where "The objective of research in information systems is to acquire knowledge and understanding that enable the development and implementation of technology-based solutions to heretofore unsolved and important business problems."

The technological solutions this research pursuit is an ontology of traceability BP, able to support the development of software solutions mainly on requirement elicitation and on solution validation [6].

This PhD work is partially conducted on a manufacturing organization, Bosch Car Multimedia Portugal S.A.. The first cycle of the research plan is currently on the development stage. The awareness of problem was grounded on lessons learned from past projects to implement traceability on the enterprise, which confirmed the negative contribution of lack knowledge to projects success, as literature also identified.

On this first cycle a traceability taxonomy is being developed. It will be used as input on next Traceability related project during requirements elicitation. On the design of current cycle a taxonomy was preferred to an ontology as the main artifact to reduce study complexity. However this option may limit study's scope to the Requirements phase of the project, as we foresee that Architecture and Verification/Validation project's phases may only be addressed on this research through the use of an ontology.

V. FUTURE WORK AND EXPECTED RESULTS

Subsequent research cycles will address the development of a Traceability Ontology and its contribution to Software Engineering on the knowledge area of Software Requirements [6]. At end of each cycle we will obtain constructs (i.e. basic language of concepts to characterize phenomena), models (i.e. constructs combined in higher order constructions), and methods (i.e. ways of performing goal-directed activities) [34]. In the process, this research may contribute on the improvement of theories related with the methodological construction of the artifacts or, related with relationships between artifact elements [23, 30].

Through the development of the Traceability Ontology we will obtain a well-organized body of organizational and strategic knowledge. To ensure that resulting ontology is generic, yet complete, major inputs for its creation will come from existing literature on academic and business field, namely existing traceability standards. This approach discards the single and specific knowledge that may exist on the development environment, in favor of the one with broad acceptance. Simultaneously, by enforcing the use of similar inputs it expectable the outcome of similar traceability ontologies. This knowledge shared across IT department and other stakeholders, will ground the deepening and sharing on the understanding [47]. This research vector with main focus on creating an ontology will use as start-up studies aiming the development of an enterprise ontology [4, 7] and product ontology [46], and on its prosecution adjust and improve the theories, methods and models used. We intend to use the 4SRS (Four Step Rule Set) method on the ontology development, and also to promote the results uniformity and quality [48].

Research cycles linked to the installation of traceability on an organization will also enable to pursue the reuse of domain knowledge [32] and the prevention of misunderstandings [20]. These research cycles focus on the ontology use, as a source of generic requirements to an IS solution supporting traceability BP, which instantiate the systematic framework conceived by Yu [47] to help developers understand what stakeholders want. As Sutcliffe [3] and Lam [32] endorsed they promote re-usability, even at later stages, improving software development productivity and quality.

Another study focus is the use of the ontology to support the verification and validation of requirements expressed by stakeholders and of the models on proposed software solution. The development of techniques that, by overlapping the ontology and stakeholders' requirements, base the evaluation of requirements reasonableness, consistency, completeness, suitability, and lack of defects [19]. We also expect that the ontology may be used (translated) as meta-model enabling the quality inspection of the software solution' models. More than behavioral models, on traceability, the data models [36] are critical due to the large volumes of information it uses and generates. Careful decisions need to be made about what information the system will need to represent, and how the information held by the system corresponds to the real world phenomena being represented.

Also on this research cycle we will study the creation of the domain model through the traceability ontology instantiation.

This research will reduce the task of creating application-specific models and will provide tools for its evaluation [39].

VI. Conclusions

Literature review enlightened that traceability is important to the scientific community, and also, that serious problems are still demanding proper solutions.

Organizations may obtain immediate benefits, if current difficulties they face when handling this business process are reduced. The root of main difficulties, lie on the lack of understanding and agreement, by main players on the meaning of traceability concepts, concrete demands, and the process nature itself. Several efforts were developed to minimize this problem through the creation of standards, laws, and regulations. Yet, each of them was unable to produce a complete traceability conceptualization or implementation guideline. Each one is focused on a strict range of interests, and scope it addresses.

This document proposes the use of software engineering methods and techniques (namely, ontologies and models) to aggregate, disambiguate, and blend existing knowledge.

This research expects to contribute to the body of knowledge of traceability business process, mainly to the software requirements community. Main relevance of this study will come from artifacts conceived and respective applicability on manufacturing organizations to implement software solutions.

The analysis and synthesis of literature on ontology building is also expected to produce a valuable feedback to respective authors, regarding completeness, coherence, etc.

The development and use of the artifacts, constructs, models, methods, and theories will be tested, and improved or adapted. The observation of this development will bring new knowledge to ontology engineering and requirements engineering.

References

[1] TRACE project, 2005. URL http://trace.eu.org.

[2] PETER project, 2006. URL http://www.eu-peter.org.

[3] a.G. Sutcliffe, N.a.M. Maiden, S. Minocha, and D. Manuel. “Supporting scenariobased requirements engineering.” IEEE Transactions on Software Engineering, 24(12):1072-1088, 1998.

[4] Antonia Albani, J. Dietz, and J. Zaha. Identifying Business Components on the basis of an Enterprise Ontology. Interoperability of enterprise software and applications, pages 335-347, 2006.

[5] APCER. Guia interpretativo NP EN ISO 9001:2008. 2010.

[6] IEEE Computer society, SWEBOK – Guide to the Software Engineering Body of Knowledge, 2004 version

[7] P. Bertolazzi, C. Krusich, and M. Missiko. An approach to the denition of a core enterprise ontology: CEO. In OES-SEO 2001, International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations, pages 14-15.

[8] Massimo Bertolini, Maurizio Bevilacqua, and Roberto Massini. FMECA approach to product traceability in the food industry. Food Control, 17(2):137-145, February 2006.

[9] Brian L. Buhr. Traceability and information technology in the meat supply chain: implications for rm organization and market structure. Journal of Food Distribution Research, 34(3):13-26, 2003.

[10] J.C. Bureau and Egizio Valceschini. European food-labeling policy: successes and limitations. Journal of Food Distribution Research, 34(3):70-76, 2003.

[11] Xin Chen. RFID Middleware Design Research. Applied Computing, Computer Science, and Advanced Communication, pages 50-56, 2009.

[12] Christian Coff, D Barling, and M Korthals. Ethical traceability and communicating food.Politics, 2008.

[13] European Parliament. Regulation No 1830/2003 of the European Parliament and of the Council of 22 September 2003 Directive 2001/18/EC, 2003.

[14] Birgit Gampl. Traceability systems in the German food industry - towards a typology. Schiefer et al2003, (September), 2003.

[15] GS1, GS1 Standards Document GS1 Global Traceability Standard. Number 1. GS1, 2009.

[16] Gabriel Hermosillo, Julien Ellart, Lionel Seinturier, and Laurence Duchien. A Traceability Service to Facilitate RFID Adoption in the Retail Supply Chain. Europe, 2009: 49-58, 2009.

[17] Gasevic, D., D. Djuric, and V. Devedzic, 2006, Model Driven Architecture and Ontology Development (Spring-Verlag, Berlin, DE).

[18] A.R. Hevner, S.T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. Mis Quarterly, 28(1):75-105, 2004.

[19] Ann M. Hickey and Alan M. Davis. Requirements Elicitation and Elicitation Technique Selection : A Model for Two Knowledge-Intensive Software Development Processes. In System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, page 10, 2003.

[20] Rudy Hirschheim and Heinz K. Klein. Four paradigms of information systems development. Communications of the ACM, 32(10):1199 1216, October 1989.

[21] Jan Holmström and Kary Främling. Design Ppatterns for Loosely Coupled Track , Trace ,Conguration , and Check Operations in Multi-company Environments. In Proceedings of EUROMA'2005 conference ,, number 1, pages 1-10, 2005.

[22] Juhani Iivari. A paradigmatic analysis of information systems as a design science. Scandinavian Journal of Information Systems, 19(2):5, 2007.

[23] John R. Venable. The role of theory and theorising in design science research. of the 1st International Conference on Design Science, 2006.

[24] M .Jansen-Vullers, C.A. van Dorp, and A.J.M. Beulens. Managing traceability information in manufacture. International Journal of Information Management, 23(5):395-413, October 2003.

[25] I.J. Jureta, John Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In International Requirements Engineer-ing, 2008. RE'08. 16th IEEE, volume 2008, pages 71-80. IEEE, 2008.

[26] Mikko Kärkkäinen, Timo Ala-Risku, and Kary Främling. The product centric approach: a solution to supply network information management problems? Computers in Industry, 52(2):147-159, October 2003.

[27] Mikko Kärkkäinen, Timo Ala-Risku, and Kary Främling. Eficient tracking for short-term multi-company networks. International Journal of Physical Distribution & Logistics Management, 34(7):545-564, 2004.

[28] M.R. Khabbazi, N. Ismail, Md. Yusof Ismail, and S.a. Mousavi. Data Modeling of Traceability Information for Manufacturing Control System. In 2009 International Conference on Information Management and Engineering, pages 633-637.Ieee, April 2009.

[29] M.R. Khabbazi, N. Ismail, Md. Yusof Ismail, and S.A. Mousavi. Modeling of Traceability Information System for Material Flow Control Data. Australian Journal of Basic and Applied Sciences, 4(2):208-216, 2010.

[30] B. Kuechler and V. Vaishnavi. On theory development in design science research: anatomy of a research project. European Journal of Information Systems, 17(5):489-504, 2008.

[31] Björn Kvarnström. Traceability in Continuous Processes Applied to Ore Renement Processes. PhD thesis, LuleåUniversity of Technology, 2010.

[32] W. Lam, J. a. McDermid, and a. J. Vickers. Ten steps towards systematic requirements reuse. Requirements Engineering, 2(2):102-113,June 1997.

[33] L. Macaulay. Requirements for requirements engineering techniques. Proceedings of the Second International Conference on Requirements Engineering, pages 157-164, 1996.

[34] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. Decision Support Systems, 15(4):251-266, December 1995.

[35] Miguel De Castro Neto, Maria Brandão L. Rodrigues, Pedro Aguiar Pinto, and Isabel Berger. Traceability on the WEB - A Prototype for the Portuguese Beef Sector. In EFITA, number July, pages 607-611, 2003.

[36] B. Nuseibeh. Weaving together requirements and architectures. Computer, 34 (3):115-119, March 2001.

[37] Pejman Oghazi. Traceability in continuous grinding circuits. PhD thesis, LuleåUniversity of Technology, 2008.

[38] Hervé Panetto, S. Bannina, and Gérard Morel. Mapping the IEC 62264 models onto the Zachman framework for analysing products information traceability: a case study. Journal of Intelligent Manufacturing, 18(6):679-698, 2007.

[39] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, 27 (1):58-93, 2001.

[40] Balasubramaniam Ramesh, Amrit Tiwana, and Kannan Mohan. Supporting Information Product and Service Families with Traceability. SOFTWARE PRODUCT FAMILY ENGINEERING, 2290:353-363, 2002.

[41] Rohan Samarasinghe, Duminda Nishantha, Noriyuki Shutto, and Manjula Wanniarachchige. Total Traceability System : A Novel System by Combination of Horizontal and Vertical Traceability Systems for Food Supply Chains. IJCSNS International Journal of Computer Science and Network Security, 9(3):148-156, 2009.

[42] Mayank Shridhar and Amit Dilip Deshpande. Supply Chain Traceability with RFID and SAP. Infosys - White Paper, 2010.

[43] Sergio Terzi, Jacopo Cassina, and Hervé Panetto. Development of a metamodel to foster interoperability along the product lifecycle traceability. In Interop-ESA, 2005.

[44] Sergio Terzi, Hervé Panetto, Gerard Morel, and Marco Garetti. A holonic metamodel for product traceability in Product Lifecycle Management. International Journal of Product Lifecycle Management, 2(3):253, 2007.

[45] Ilkka Töyrylä. REALISING THE POTENTIAL OF TRACEABILITY - A case study research on usage and impacts of product traceability. Dissertation for the degree of doctor of technology to, Helsinki University of Technology, 1999.

[46] Angela Tursi, Michele Dassisti, and Hervé Panetto. Products information interoperability in manufacturing systems. Annual Reviews in Control, 33(2), 2009.

[47] E.S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering, pages 226-235, 1997J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[48] Nuno Ferreira, Nuno Santos, Ricardo J. Machado, and Dragan Gašević, “Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design”, PROFES, Vol.7343 Springer (2012), p.44-58

A Software Framework for Supporting Ubiquitous Business Processes: An ANSI/ISA-95 Approach

Manuel João Amaro
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
manuel.amaro@algoritmi.uminho.pt

Ricardo J. Machado
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
rmac@dsi.uminho.pt

***Abstract* — Nowadays, organizations to survive competitively they need to be, innovative and efficient. The way the Internet has been expanding along with other technological changes is leading us to a future in which all the objects that surround us will be seamlessly integrated into information networks. The possibility to implement concepts related with the ubiquitous computing in the business process-level will influence how they are designed, structured, monitored, and managed. One of the most remarkable possibilities of ubiquitous computing can be the real-time monitoring of a particular business process: it should be possible to analyze the flow of materials and information, identify possible points of failure or improve energetic efficiency with a small delay on they occur in reality. Currently, there is no direct and automated link between ubiquitous business processes descriptions and their physical executions which, frequently, promotes the occurrence of a discrepancy between the planned modes of operation and the executed ones. The ubiquitous business processes will enable a narrowing between the real (objects) and virtual (models) world and the possibility to create adaptive business processes that can predict failures, adapting themselves to changes in the environment is an attractive challenge. In this PhD thesis, we will propose a new software framework to monitor real-time executions of ubiquitous industrial business processes.**

Keywords: software design, computer science related discipline: information management, management related discipline: information systems management

I. INTRODUCTION

The first reference to ubiquitous computing dates from 1993, when Mark Weiser projected the future as he imagined: "The idea of ubiquitous computing first arose from contemplating the place of today's computer in actual activities of everyday life" [1]. Since then, there have been tremendous developments in technology, many new concepts have appeared, others suffer various changes, however Mark Weiser's words still hold true. The effort expended in the study of ubiquitous systems and technologies that support them has gained considerable interest and has been the target of several advances, whether in academic or in industrial fields. In another study [2] is also noted that "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it".

In addition to the benefits inherent from the technological advancement and deployment of ubiquitous computing, users of these systems should be key elements, acting, interacting and improving these environments. Ubiquitous computing is more than just allowing the various devices (in a common environment) to communicate/interact among them. It also consists in the way they do it, in the way they interact with users and how they can help users achieving their goals [3].

The user satisfaction is a key element for the success of ubiquitous systems; however, the use of ubiquitous computing not always aims the satisfaction of a single user, but the satisfaction of an organization, a group of people, a value chain or a business. The use of ubiquitous computing in organizations has been growing, not only for employee satisfaction, but also to improve work methods, processes, efficiency, to reduce production costs, etc. Ubiquitous computing has the capacity to improve the way business processes are (re)designed and executed, which in turn will bring more competitive companies and better economy efficiency by allowing these same companies to invest more in technology development [3]. It is therefore a win-win situation, if applied properly.

In organizational contexts, it is frequent the occurrence of discrepancies between the planned modes of operation of business processes and the executed ones. As an example, in an industrial company we may need to monitor and manage in real-time the production status. This monitoring and management tends to be difficult when recurring to current business processes models, because their observation relative to the real operations is done long after their executions and thus important data may be missing as well as reaction time may be surpassed. There is a big time delay in the perception of problems, by management teams, and also delays in reacting to them, which can lead to serious problems and costs that should be avoided. Additionally, the human-reported data tends to be not so accurate and with lower scope as data collected directly from the business process elements. This data gaps endanger the ability for organization to manage based on concrete and detailed facts.
Without the constant monitoring of business processes, in all their execution phases, it becomes impossible to manage them adequately. Hence, it is necessary to investigate how to

978-0-7695-4777-0/12 $26.00 © 2012 IEEE
DOI 10.1109/QUATIC.2012.18

benefit from the ubiquitous computing principles to support the monitoring of business processes. It is expected that this new approach would influence how business processes are designed, structured, monitored, and managed. We need to adequately design all the concepts to trigger, store, and manage all the data relative to the business process execution. This design should be a consequence of each organization requirements and can be materialized on traceability concerns and policies deployed into the design and execution of business processes. We need a software framework to design solutions capable of linking these two worlds.

II. STATE-OF-THE-ART

To support the ubiquitous computing vision that Mark Weiser projected in 1993, beyond any technological evolution, innovation must continue because the paradigm of ubiquitous computing is different from traditional computing paradigms. Imagine a closed environment where all objects present have the ability to communicate; a ubiquitous system that supports this environment will have to bear not only the heterogeneity objects but also the possible different forms of communication. If it is already difficult to conceive the ubiquitous system for a closed environment, imagine for an open environment, where the objects appear and disappear randomly. In these types of systems we now have an environment not only diverse but also decentralized, where various types of objects communicate using different technologies. The benefits inherent in the use of ubiquitous computing were readily assimilated by organizations that seek to optimize their processes, to reduce costs, and by organizations that want to continuously improve and generate profit. Strassner [4] argues that “when companies plan to adopt a new technology, they want to know the business impacts in advance”. This capability allows companies to better control their processes, avoiding harmful situations (e.g. the bull-whip effect), thereby improving the flow of material, the flow of information, eliminating the production to stock, and excess production. Supply chain inefficiencies can waste up to 25 percent of a company’s operating cost [4].

The bull-whip effect is very harmful to any organization, and the use of “pillows” such as the creation of stocks is not certainly the best strategy because it increases the company costs. It is in cases like these that the ubiquitous systems have a role to play, arming organizations with ubiquitous processing power, allowing them to create a harmony within the value chain, and coordination between the flow of materials and the flow of information.

Recently, Lupiana [5] proposed a taxonomy to distinguish ubiquitous environments. He categorized UbiComp environments in two major classes: Interactive and Smart environments. In turn, Chen proposes an ontology (SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications” [6]) for the creation and development of ubiquitous/pervasive applications.

Regarding the development of applications for pervasive information systems, it should be noted the article “Model-Driven Methodologies for Pervasive Information Systems Development”, where the authors report that “we become aware of the presence flow and processing of information, not only by the individual computing devices, but also with a more deep significance, by the overall system that emerges from the interactions of all the computing devices, linking them together in a coherent fashion” [7], which values the need to have a holistic view.

A. Business Processes

In day-to-day activities, organizations interact with multiple and distinct entities. These entities can either be part of the internal organizational structure (e.g. in case of large organizations) or external agents to the organization, but playing an important role, such as suppliers, customers, etc.

For all these entities to function properly and in harmony it is necessary to establish processes, tasks and activities so that everyone can work with a common goal. Good communication is a key element to the various entities that communicate with each other. It was based on these assumptions that business processes arose. They plan to serve a set of processes, tasks, and activities that must be carefully performed by various entities, at indicated times and in a specific order. The main objective of a business processes manager is to have a holistic view of the entire organization (from suppliers and raw material to customers and finished product) in order to define a set of processes that aims overall improvement, cost reduction, waste reduction, reworking, and productivity. The processes of an organization reflect the way tasks and procedures are performed, and can (and should) be redesigned whenever possible in order to ensure continuous improvement.

A business process is triggered by a business event, and aims to delineate a set of procedures/activities to be performed by people, machines and/or computers. These participatory elements have distinct roles and objectives throughout the process course. A process consists in a specific order of work activities across time and place with a beginning, an end and clearly identified inputs and outputs [8]. When modeling business processes, we need to take care of several items. Two key issues are how much detail and how to handle uncertainties. The level of detail will allow us to know how deeper we want to go when decomposing the process. The way we expect/control uncertainties, will allow to have mechanisms to control the process. The uncertainty is one of the main reasons why the procedures deviate from what was previously stipulated [9].

Monitoring consists in collection, compilation, analysis and presentation of data that reflects how a particular business process is being executed and managed by different agents (from people to machines). It is easily verifiable that trough monitoring one can know if a process is or is not being properly executed, or whether it is well or poorly modeled. The real-time monitoring allows for timely decisions that can prevent a future malfunction of the processes. A monitoring system observes the behavior of other system and checks if it is consistent with what is expected, with a given specification [10].

The topic of real-time monitoring is very controversial, starting immediately from the definition of the word "real-time".

B. *Ubiquitous Business Processes*

Again, we emphasize that the ubiquitous systems have a very important role in monitoring business processes. With the use of smart items, we can follow the state of the products. Huang [11] developed and implemented a practical solution for monitoring business processes using ubiquitous systems. In this particular case were used RFID tags to easily monitor the inventory of an organization. He analyzed and proposed the best places to put the antennas that would read the RFID tags, where to place the tags, etc, in order to have a real-time inventory management.

In another research, Zhang [12] describes a smart kanban system using RFID technologies for shop-floor management and several relevant real-time manufacturing data capturing cases using RFID, wireless production lines and wireless shop-floor inventory management. In this particular case, the use of RFID tags within kanbans is a very interesting concept because it allows monitoring a lot of articles along the chain at a low cost, because the RFID tags are reused since they are an integral part of the kanban.

A practical example of a monitoring system was developed and presented by Huang [11], which proposes an infrastructure to implement RFID for product tracking and monitoring. They also propose formulas for comparing the actual state of the process compared with planed one.

III. RESEARCH OBJECTIVES AND METHODOLOGICAL APPROACHES

A. *Research objectives*

The everyday work in a large automotive company, allows us to experience and witness certain events which sometimes raise doubts and questions that are not easy to answer. This research proposal follows exactly this perception from the reality and of the experienced events, which, inserted into a dense organizational structure, sometimes becomes so obscure and dissimulated, that are unnoticed by who has the capacity to take decisions and is responsible for the design of business processes, which are so vital to any company.

The ANSI/ISA-95 standard [19] consists of models and terminology, and addresses the issue of integration of different information systems, software applications, Programmable Logic Control(lers) (PLC), etc, in industrial systems. The ANSI/ISA-95 is divided into 5 parts: (part 1) models and terminology (published 2000); (part 2) object models and attributes (published 2001); (part 3) activity models of manufacturing operations management (published 2005); (part 4) object models and attributes of manufacturing operations management (under development); (part 5) business to manufacturing transactions (under development).

This standard plays a critical role in our research, since we will adopt an approach compliant with the part 4. We will identify which kind of information needs to be exchanged across the several organizational layers for user requirements purposes and also for database and software development. Our software framework will allow the design and execution of interfaces, assuring the correct flow of information between the enterprise information system and the manufacturing operations system.

Fig. 1 illustrates the different structural/hierarchical levels existing in a typical industrial organization. The ANSI/ISA-95 standard organizes the different organizational layers and provides standard protocols for exchanging information, thus facilitating the flow of information through the different layers of the organization:

- In level 4 (business related activities), activities are not directly related to production (long term planning, marketing, sales, procurement). At this level we can find ERPs, where business and logistics planning occurs.

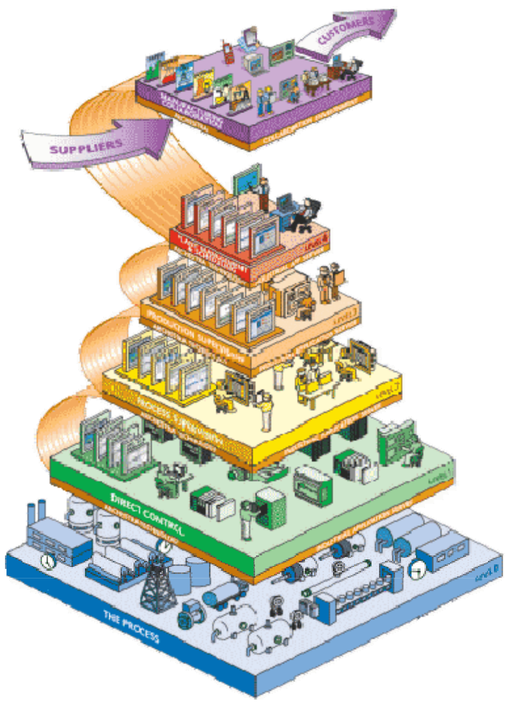

Fig 1: ISA Model (adapted from [18])

• In level 3 (manufacturing execution system) work flow activities produce the desired end products. Consists of several activities that must be executed to prepare, monitor and complete the production process that is executed in the lower levels (0 - activities of monitoring, supervisory control and automated control on the production process; 1 - activities involved in sensing and manipulating the physical process; and 2 - actual physical process). Examples are: detailed scheduling, quality management, maintenance, production tracking, etc.

The time frame adopted in level 4 can be in days, months or even years. As long as we go down into the levels, the time frame changes to days, shifts, hours, minutes or seconds. At level 2, the time frame can be sub-seconds or milliseconds. Within this PhD work, we will address levels 3 and 4, which relate to business processes. We will also address level 2, in order to interoperate ERPs to the PLCs, in order to decompose the business processes and monitoring them at level 2. Our research objectives are:

• Based on models of existing business processes (requirements specifications from the software point of view), we intend to develop a method to decompose the business processes allowing the definition of where and how real-time monitoring should be done. This method will also enable the formalization of the ubiquitous nature of the business processes.

• Based on concepts from the ubiquitous computing paradigm, we intend to develop a software framework to support the monitoring of the industrial ubiquitous business processes executions. We will adopt behavioral and architectural patterns to support the interoperability of sub-systems.

• To validate the proposed method and framework in real scenarios of industrial production environments from Bosch Car Multimedia factory plant.

B. *Methodological approach*

Design Science Research (DSR) [13] will be adopted as the main research method. One of the main reasons that led to the choice of this method is that its main objective resides in solving real problems. DSR is a normative and prescriptive method, and the researcher is usually pragmatic. DSR is elaborated trough the relationship between two main activities: build/design and evaluate, where researchers recur to kernel theories in order to develop artifacts and then demonstrate that they can be built [14]. These kernel theories frequently derive from disciplines outside of information systems area, and suggest novel techniques or approaches to IS design problems [20].

This model (Fig. 2) of "construction" and "evaluation" has been used in the past to develop new knowledge through the construction and performance evaluation of new artifacts.

Development, *Evaluation*, and *Conclusions* stages can be iterative; i.e., the result of each stage can trigger the start of another cycle. The *Circumscription* process is especially important because it generates understanding that could only be gained from the specific act of construction; it assumes that every fragment of knowledge is valid only in certain situations [13].

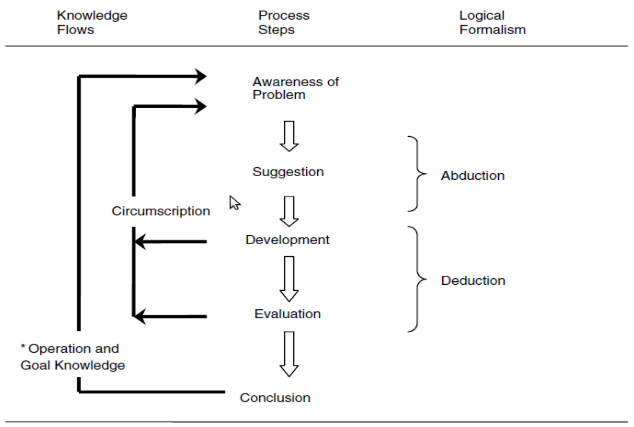

Fig. 2: The General Design Cycle [13]

Once the artifact is finished, it has to be returned to his environment in order to be studied and evaluated in their application domain [15]. For Hevner the development of an artifact relies on kernel theories, which are applied, tested, modified, and extended through the creation of artifacts. Although based on existing theories, DSR also contributes to the "construction of new theories" or to the improvement of existing ones [13]. Better theories are one of the possible outputs of DSR, as shown by Hevner and Chatterjee [16].

DSR always seeks a solution to solve practical real world problems, and IT artifacts are the end-goal of any design science research project, and are broadly classified into: constructs (vocabulary and symbols); models (abstractions and representations); methods (algorithms and practices); instantiations (implemented and prototype systems); better design theories.

Vaishnavi considers constructs as the conceptual vocabulary of a problem/solution domain and they appear during the conceptualization of the problem, and models as a set of propositions or statements expressing relationships among constructs. Methods are objective oriented, and consist of a step of steps used to execute a task. Finally, instantiation is the next step, the accomplishment of the artifact in an environment [13].

Samuel-Ojo [17] refers that "research in the information systems field examines more than just the technological system, or just the social system, or even the two side by side; in addition, it investigates the phenomenon that emerges when the two interact". It is not easy to design useful artifacts; sometimes the researcher needs to be creative because existing theories in those specific domains areas are insufficient. These artifacts have to create direct impact on organizations and in society in general.

IV. PAST WORK AND PRELIMINARY RESULTS

We have already been involved in the creation of real-time monitoring systems at Bosch Car Multimedia plant. We have adopted the DSR approach in two projects whose aim was to develop systems for real-time data acquisition and information delivery. The acquired data was also used to trigger alarms and measure real-time reaction times.

The *Milk Run Realtime Information* (the first DSR execution) consisted of a real-time monitoring system for the supply convoys of the manufacturing (manual and automatic) production lines. This project meant to monitor the supply time of production lines, by the convoys. Monitoring was done upon leaving the warehouses, and controlling was carried out in various points of the route. Whenever time deviations occurred in relation to the planned, alarms were automatically triggered and the reactions times (by interventions teams) were also calculated and stored for future use.

In the *Realtime ANDON* project (the second DSR execution), a solution for real-time monitoring of manual insertion lines was developed. These lines worked for 24 hours a day, in 3 shifts. To accomplish this project, several points of control along the production line were adopted, by using SOAP protocol to communicate with the central server, informing the location and identification of the products. We have used infrared technologies to read the serial number of the unit. Taking into account: (1) the production time of each product (a line, during a shift, may produce several types of products with distinct production times); (2) the production plan defined, for each shift; and (3) the production pre-defined breaks; we have been able to calculate, in real-time, the production rate of the line and also to show, either in time or units, the delay/advance checked against the planned/expected production plan.

Both solutions caused impact on the organization, and helped, not only to find faults, but also to help improve the production plan. In the second project, it should be noted that on the beginning, it was received with some reluctance from the production line workers, because their work was being monitored and measured constantly, and published in real-time to everyone in the factory trough big TV screens. However, we found that the people themselves began to use the system in order to know, in real-time, the amount of unities they had to produce to finish the shift, and to compete with neighboring production lines. Using the data collected by this monitoring system, it was also possible to recalculate the production times of several products and automatically calculate the quick change over (QCO) between products, and thus improve the production planning.

In both projects open source software (like Apache, MySQL and Linux) was used. The programming languages used were Perl and PHP. The communication protocol used between the control points and the server was SOAP. During the execution of the first DSRs, the experience with all these technologies for implementing those two projects has enabled us to get real knowledge about the problem domain and also to start the effort to design the software framework that will be further developed and explored in future projects.

V. FUTURE WORK AN D EXPECTED RESULTS

In the near future, we will start the formalization of the software framework based on the first two DSR executions to enable the adoption of new mechanisms in the next projects to be developed. Only after the formalization of the first perception of the software framework we will be able to come-up with some new behavioral and architectural patterns and also with some technological insights that we may discover to be innovative and efficient. The process of literature review will allows us to base the writing of our scientific contributions on solid arguments. We intend to publish our contributions in reputed journals/conferences: EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA); International Conference on Information Systems Development (ISD); AIS European Conference on Information Systems (ECIS); International Journal of Computer Integrated Manufacturing (IJCIM); and International Journal of Enterprise Information Systems (IJEIS). Publication and/acceptance on this set of journals and conferences aims to validate the research quality, the quality of the articles, the scientific contribution and the recognition of the developed work.

VI. CONCLUSION

The importance of modeling business processes in any company is of utmost importance nowadays. With recent technological advances, such as the ubiquitous computing, the potentialities from these advances may come, in some way, to affect the way business processes are modeled, monitored, and executed. Ubiquitous computing, and all the advantages/disadvantages that come with it, are not yet completely known in the industrial world. Despite not having much work done in the field, there is already some knowledge and concern over this issue.

One of the main concerns identified, resides precisely in the absence of real case scenarios with economic impact in organizations. This is actually one of the points to improve, in order to change some existing paradigms in organizations, opening new horizons and promoting innovation; otherwise they risk being overtaken by competition.

Ubiquitous computing will then empower organizations with new resources, which may, when properly used, help to monitor their industrial processes, providing relevant information to whom right, in real-time. For it to become real, it is necessary to cut the existing business processes, i.e., decompose them, in order to know how and where monitoring should be done and, more importantly, how and by whom, by which decision agents. It is necessary to formalize the ubiquitous nature of the business processes.

This new understanding of formalization of the ubiquitous nature of the business processes, and how they must be implemented and executed, allows the creation of a software framework that can monitor real-time executions of ubiquitous industrial business processes.

This proposal seeks to validate this software framework, in real settings, by practical application, because its contributions, if advantageous, could be directly incorporated into companies' best practices. The proposed research method, DSR, fits gracefully in those, which are the premises of this research. We propose to find a practical solution to real problems, by building IT artifacts, in this case, a software framework.

REFERENCES

[1] Weiser, M. (1993). Some Computer-Science issues in Ubiquitous Computing. Communications of the ACM, 36(7):75–84.

[2] Weiser, M. (1995). The computer for the 21st century. Scientific American, 272(3):78–89.

[3] Fernandes, J. E. M. (2010). About Model-based Approaches in pervasive information systems development.. PhD thesis. University of Minho

[4] Strassner, M. and Schoch, T. (2004). Today's impact of ubiquitous computing on business processes. Institute of Information Management of University of St. Gallen.

[5] Lupiana, D., O'Driscoll, C., and Mtenzi, F. (2009). Taxonomy for Ubiquitous Computing Environments. ieeexplore.ieee.org, (1).

[6] Chen, H., Perich, F., Finin, T., and Joshi, A. (2004). SOUPA: Standard ontology for ubiquitous and pervasive applications. The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004., pages 258–267.

[7] Fernandes, J., Machado, R., and Carvalho, J. (2004). Model-driven methodologies for pervasive information systems development. MOMPES'04, page 15.

[8] Lindsay, a. (2003). Business processes—attempts to find a definition. Information and Software Technology, 45(15):1015–1019.

[9] Kirkwood, C. W. (1998). System Dynamics Methods. College of Business Arizona State University USA

[10] Peters, D. (2002). Requirements-based monitors for real-time systems. IEEE Transactions on Software Engineering, 28(2):146–158.

[11] Huang, G. Q., Zhang, Y., and Jiang, P. (2007). RFID-based wireless manufacturing for real-time management of job shop WIP inventories. The International Journal of Advanced Manufacturing Technology, 36(7-8):752–764.

[12] Zhang, Y., Jiang, P., Huang, G., Zhou, G., and Zhao, L. (2010). RFID-Enabled Real-Time Manufacturing Information Tracking Infrastructure for Extended Enterprises. In Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology, pages 1723–1734. Springer.

[13] Vaishnavi, V., Vaishnavi, V., and Kuechler, W. (2007). Design science research methods and patterns: innovating information and communication technology. Auerbach Pub.

[14] Lind, M., Rudmark, D., and Seigerroth, U. (2010). Design Science Research for Business Process Design: Organizational Transition at Intersport Sweden. Human Benefit through the Diffusion of Information Systems Design Science Research, (Matthews 2007):159–176.

[15] Hevner, A. (2007). A three cycle view of design science research. Scandinavian Journal of Information Systems, 19(2):87–92.

[16] Hevner, A. and Chatterjee, S. (2010). Design Research in Information Systems. Information Systems Research, 22:9–22.

[17] Samuel-Ojo, O., Shimabukuro, D., Chatterjee, S., Muthui, M., Babineau, T., Prasertsilp, P.,Ewais, S., and Young, M. (2010). Meta-analysis of Design Science Research within the IS Community: Trends, Patterns, and Outcomes. Global Perspectives on Design Science Research, pages 124–138.

[18] http://www.intellient.co.za/live/content.php?Item_ID=475

[19] http://www.ansi.org/

[20] Kuechler, B. and Vaishnavi, V. (2008). On theory development in design science research: anatomy of a research project. European Journal of Information Systems, 17(5): 489-504

2012 Eighth International Conference on the Quality of Information and Communications Technology

Modeling Organizational Information System Architecture Using "Complex Networks" Concepts

José L.R. Sousa
Sistemas de Informação
IBMC
Porto, Portugal
jsousa@ibmc.up.pt

Ricardo J. Machado
Centro ALGORITMI
Universidade do Minho
Guimarães, Portugal
rmac@dsi.uminho.pt

J.F.F. Mendes
Group of Complex Systems &
Random Networks
Aveiro, Portugal
jfmendes@ua.pt

Abstract **— Organizations live in a world where interdependence, self-organization and emergence are factors for agility, adaptability and flexibility plunged into networks. Software-based information systems go into a service oriented architecture direction and the same goes to Infrastructures where services are become structures available in networks. Inspired into empirical studies of networked systems such as Internet, social networks, and biological networks, researchers have in recent years developed a variety of techniques and models to help us structurally understand or predict the behavior of these systems. Those findings are characterized by been supported on the "complex networks" concepts. On this PhD research we present the use of the concepts of complex networks from physics to develop organizational information system architectural models, as requirements modeling technique. The research is about the structure and function of networks and its use for modeling organizational information systems architectures by using a combination of empirical methods, analysis, and computer simulations.**

Keywords: software requirements, computer science related discipline: information management, management related discipline: information systems management

I. Introduction

Organizations live in a world where interdependence, self-organization and emergence are agility, adaptability and flexibility factors plunged into networks. It is a networked composed world in the design of collaborative-networked organizations. This networked structuration comes to the composition of complex systems, from cells, to society and enterprises (associations of individuals, technology and products). In those complex systems, characteristics of emergence, order and self-organization [1], develop a set of network interdependent actions not visible in the individual parts. This is what complex systems are about and networks concepts are becoming a common approach to describe and quantify these complex systems structures. A fundamental concern is to know the anatomy of theses structures in a relation that structure always affects function [2].

Organizational information system concentrates a set o elements that due to its role and nature are becoming similar to energy and raw materials, this is, are fundamental elements for organizations needs and successful existence. Developed in order to support the organizations processes, which are the center of organization efficiency and effectiveness, but without in-built value [3]. Organizational information system integrate commodities like computation power, whose current availability [4], is no longer a differentiation. In presence of this elements and in consequence, information systems are each more seen as a commodity [5] where organizational information systems are consumed by the idea of not being aligned with the enterprise information needs [7].

The realm of organizational information systems is the confluence between context, business and software-based technology and infrastructures. Organizational information systems and behavior are not dichotomous but inseparable [6] building a interdependent techno-social system. The use of complex network characteristics to model the information flow in organizational information systems is a tentative to better understand their techno-social nature and, thus, learn how to design better software-based organizational information systems architectures.

Present research is about the structure and function of complex networks and its use for modeling organizational information systems, studied using a combination of empirical methods, analysis, and computer simulations. The goal is to find the complex network structure for real organizational information systems architectures and produce models that can be analyzed and studied to better understand the information flow in the organizational techno-social system.

In this paper, a description of the state-of-the-art related with the subject of this research is presented in Section 2. Section 3 describes in detail the research objectives and the methodological approach. In Section 4, past work and preliminary results, already done in the context of this research, are briefly described. Section 5 presents future work and expected results, for the next 2 years of research. Finally, in Section 6 some conclusions are presented.

II. STATE-OF-THE-ART

Organizational information systems, are consumed by the idea of not being aligned with the enterprise information needs [7], whose existence and development is sustained by integrating a set of commodities. Organizational information systems should be planed, design developed and managed according to the organization strategy [8]. This approach makes information systems clearly responsive. There is a real need for information systems, with ability to support emergence, self-organization that are present and can support the organizations evolution [9]. Organizations are real world entities with dynamic elements, users, in space and time.

Information systems must able to support the evolution of the interdependence between the techno and social systems. The construction of this techno-social system can be modeled by complex interactions [9] that develops co-evolution trough a flexible, adaptable and agile information systems architecture structure.

A. Information systems

Defined by [10] as an "*organizational system that consists of technical, organizational and semiotic elements which are all re-organized and expanded during ISD (information systems development) to serve an organizational purpose*". According to [11], "*information systems is what emerges, from the usage and adaptation of the IT (information technology) and the formal and informal processes of all of its users*" .

Information systems must be constantly adapting to needs as the users change its use and the IT is updated or extended [11]. Information systems are also seen as communication system used to support a given human activity system, processes [12]. Information systems are a natural consequence of the need for humans to communicate and coordinate their activities [13][14]. An organizational information system, on this research proposal, is the complex network of interactions of software-based information systems (that are combinations of computation, communication, technology and processes) developed to fulfill enterprise goals and integrated in a certain context. This software-based information system is seen as SaaS (Software as a Service) anchored in a IaaS (Infrastructure as a Service) where context is the set of elements (Users) that interact with information technology. This complex networks approach to model information systems develops a architecture umbrella that is a tentative to understand the intrinsic nature of information flow and allows the design of better software-based information systems that expectably fully integrates emergence and self-organizing behaviors.

B. Commodities

Commodities lead to the concept that itself an information system does not add value. From technology [5], communications and computation [4, 15], information technology solutions are seen as a commodity. In this perspective information technology will be available at the enterprise ecosystem and can be easily plugged in and out, like energy. As an example, when a certain active entity integrates the ecosystem gets energy and everything else, including information technology. In this sense, service science is clearly developing this commodities deployment. So, this development of reality is demanding for a change that is represented in a particular part of any system, but emerges or results from a self-organization and interdependent evolution. It needs a view to develop information system architecture able to accommodate this new dynamics.

C. Complex adaptive systems (CAS)

The paradigm used to model and fundament organizations development has been changing. This change can be found in a fundamental core of articles and books that deal with this new organization dynamics [8, 18-20].

In this context, information systems architecture as a fundamental interdependent structure, develop trough an reductionist vision, will never have relevance due to its vision of parts, instead of the hole. Complexity theory and CAS can represent a response to the paradigm shift in order to address this new dynamics [21-23].

CAS integrates the concept of emergence from which adaptability and evolution arise as a result. For modeling this, complex networks are used to model self-organization, preferred attachment and fitness. Organization development is supported, by the ability to use information flow as a source for unique results, when facing change and competition along space and time [16, 17].

Complex adaptive systems are systems with great number of components, sometimes called agents that interact, adapt and learn. Many contemporary problems are under the complex adaptive system's theory [22-25]. In this systems emergence occurs near or in the limit of thermodynamic equilibrium. Such systems are common on the physical world and have "*emergent*" properties that result from interactions and are global or collective.

Emergence is founded in the existence of a global behavior that is new when related to all parts that compose the system, something not understood with the reductionist vision of the information system architecture.

Information flow then defines emergent patterns that can lead to adaptation, a familiar form in the biological process. Reorganizing genetically material, through which organism evolves to survive in environments that confront them. This process allows the modulation of non-linearity that comes from complex interactions [26, 27].

D. Complex networks

There are often cited examples of complexity, such as the Internet, WWW, immune system, ant colonies, economic markets or human social networks. Despite this fact, there is no central definition for complex systems, informally seen as a large network of relatively simple components with no central control, in which emergent complex behavior is exhibited [28]. This behavior is hard to define, and roughly, emergence refers to the fact of systems global behavior is not only complex but arises from the collective actions of simple components to which the notion of non-linearity is important: the whole is more than the sum of the parts [28].

Networks are everywhere, from brain, to society passing by organizations. Using Karl Popper [29] approach, in higher degree, this pervasive presence of networks is a construction of human mind. Internet and WWW are the most impressive creation in the information system domain and probably the most moving creation of our civilization [30]. It is possible to imagine the past and the future without them; but for the e-generation it is not [8]. It is an element present in the daily-life and from which we know little, beginning on their complex organization structure or global topology. The understanding of the Internet and WWW inherent problems is not a topic for social sciences, computer and applied mathematics but rather of non-equilibrium statistical physics

[30-32]. For properties observation of the network data is the starting point [33, 34] and the same can be argued for organizational information systems. The study of how information flows and its support under software-based information systems interactions should be done with the same topological approach, which regards the big data that is stored.

Network's structure's study started in the mathematical study of graph theory [30]. In the beginning this theory seized to Poisson distributions, resulting from simple random graphs. Moreover, by definition, random graphs in graph theory, are graphs with Poisson distribution of connections [30]. By definition "*a network is simply a collection of nodes (vertices and links (edges) between nodes. The links can be directed or undirected, and weight or outweighed"* [28].

As first stage, all networks seemed random but, along the development analysis, some different and fundamental key characteristics were found. Networks are characterized by the way in which they are created, resulting into constructs such as, degree of distribution, average path length between node's pairs, clustering degree and communities [28, 31].Barabási and Albert (echoing the earlier work by Price and others) conclude that their simple "*growth with a preferential attachment*" mechanism, is what drives the evolution of real-world networks [28]. Network theory has been used to characterize a different set of systems. This use is making network theory a strong tool for using when emergence, self-organization, dynamic and co-evolution are characteristics to be analyzed in the systems.

E. Relevant information system architecture planning models

Due to the increasing in size implementations of organizational information systems, logical models (*or architecture*) for defining and controlling the interfaces and integration of all the system components were developed. Following, two of the most relevant are presented and a discussion is made about its bottlenecks in supporting organization techno-social systems information flow.

ZACHMAN Framework: It provides a view of the subjects and models needed for complete developing or/and documenting of organization architecture [35-37]. This framework provides a basic structure that supports the organization, access, integration, development, management and changing with a set of architectural representations of organization's information system. It uses a matrix structure of 30 cells and five perspectives of the overall architecture with six classifications of the various artifacts of the architecture as well as flow diagrams. For each cell of the matrix the documentation type is suggested, using ER technique for modeling the data description or using functional flow diagrams for modeling the process description [37]. In this framework, an organization has a whole range of diagrams and documents representing different aspects or viewpoints that can be developed. In the extended framework for information systems architecture there is a meta-model for cell data, and a classification of data, process or network is made. It has no specific associated methodologies and only a set of major principles and rules exist, working as a guide. Nothing is said about processes development for viewpoints or the associated order. It represents information technology and not information systems architecture interdependence in the organizational techno-social system.

TOGAF: The Open Group Architecture Framework (commonly known as TOGAF) is an industry standard architecture framework that may be used freely by any organization wishing to develop enterprise architecture descriptions for the use within that organization. It is defined as a comprehensive architecture framework and methodology, which enables the design, evaluation and implementation of the "right" architecture for an enterprise [38], supported by a set of well-defined tools. It is composed of three fundamental parts [39]: the ADM, the Enterprise continuum, and the Resource base. ADM (Architecture Development Method) forms de core structure for TOGAF, being able to detail procedural models in order to develop descriptions of enterprise architectures. It describes the different type of inputs and outputs but does not show guidelines; ER – entity relation - is used as a formalization model [39]. Design for development and not for exploitation of techno-social systems interdependent interactions in the road to co-evolution.

III. RESEARCH OBJECTIVES AND METHODOLOGICAL APPROACH

A. Research Goals

The present research is about the discovery of big data information flow structured under complex networks, and its use for modeling organization information systems architectures, using a combination of empirical methods, mathematical analysis, and computer simulations. These computer simulations will address the visual modeling of the network structure. From its review and results constructs of the complex network theory such as clustering, path-length, degree and communities, develop the ability to model emergence, self-organization, evolution and dynamic through space and time of the organization techno-social system

Regarding this, the main goals for PhD work are:

- Define an approach to support the adoption of the complex network meta-model to analyze models of already existent organizational information systems, based on information technology big data information flow.
- Adopt those models for monitoring the dynamic execution of existent organizational information technology systems.
- Based on the monitoring results, analyze the corresponding characteristics of the organizational information system architecture.

B. Research Approach

In order to validate this approach, is adopted a positivistic research method trough a quantitative data collection. Later on, an observing interpretation will be needed and a qualitative vision will be adopted, supported in the use of and abductive inference.. All the development phases will be based on action research theory [40]. The use of action research theory comes from its use on information systems research, when there is the need to determine the complex or diverse nature of information systems. The complexity support and diversity is clear in the modeling goals of our research, so the choice for action research is adequate. Traditional approach to the definition of the action research methodology is a spiral of interactions through time. At each cycle a more close set of interaction can be implemented [40]. For the present research proposal a more simplified version of action research will be used: the demonstration cases. This approach adopts a single cycle for data collection in one selected organization.

Information system architecture is a world of big data logs and traceable information flow interdependence in the techno-social organization system. There are many tools for collecting that information. In this research, standard tools will be used and it is expectable that a plethora of models will emerge from big data [41] evaluation using quantitative modeling within complex networks constructs. A qualitative approach will also be present in our efforts for the interpretation of the results obtained from the positivist construction of models using an abductive inference for its predictive fitness. Starting from the presented concepts in this paper, a more profound literature review will be done, exploring fundamental concepts from theoretical physics that will be used and also trough the evaluation of users behavior relating to technology and predictive fitness. This combination of research methods is a great enrichment for our research approach. The global research process will be centered in three main phases, described in Fig. 1.

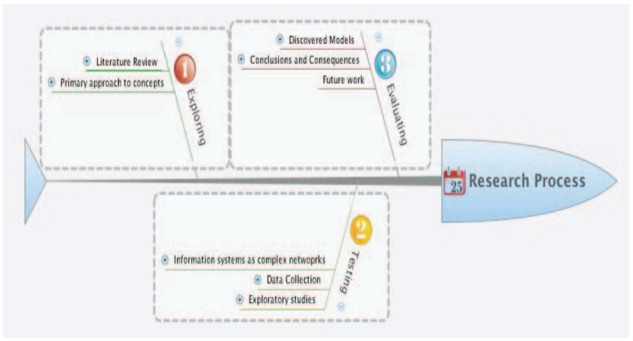

Figure 1 - Overview of the research process.

This research process starts with a broad question: *How already existent organizational information systems can be characterized adopting the "complex networks" concepts?*

Along our PhD work, three organizations will be involved for performing data collection and analysis supporting the construction of *"complex networks"* models. Two of these organizations are from two relevant industrial sectors in Portugal: automotive and textile. This phase will present modeling results of the application of complex network concepts to the organizational information systems using demonstration cases.

IV. PAST WORK AND PRELIMINARY RESULTS

This PhD work takes place within the Software Engineering and Management Group (SEMAG) from the ALGORITMI Research Centre at the University of Minho. SEMAG research group is devoted to study the development process of software-based information systems and related methodologies, focusing on both the engineering and management aspects. We will adopt a three-layer techno-social approach to regard into the commoditization process of enterprise information systems architecture in a foundation of service science (see Fig. 2): [42](1) the user level; (2) the SaaS (software as a service) level; (3) the IaaS (infrastructure as a service). We will manage a set of experiences (real big data collection and analysis) in order to produce the complex network models for each of the three levels.

A first data collection experiment was put in place at IBMC addressing the SaaS level- the level were processes are translated into software based packages - using a small link with a volume 40 to 50 connected users and *Allot NetEnforcer Series AC-500* equipment (with the cooperation of Palo Alto Networks[1]); a commodity solution that can collect data flows. The data was collected during a three-week period and then exported to CSV files through the use of *Netxplorer*, to be processed by external software for complex networks evaluation..This resulted in the construction of the adjacency list of the networks [33]. The software used for complex networks structure discovery and constructs evaluation was *Gephi.*

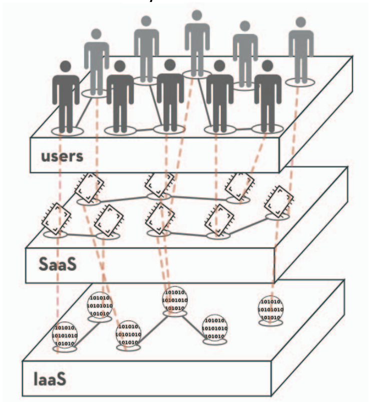

Figure 2 – 3-layer information systems modeling architecture.

The experienced was conduced with the guarantee of total confidentiality, since the data flow is only related within the IP address and additional information is needed to correlate. The architecture of data-collecting infrastructure is presented at Fig. 3. It was performed in a five-flours building in a LAN with a clear DMZ and core switching linking all the distributed connections. The data collection allowed us to discover initial structures of the information flow that are the core element of the "complex network" model.

[1] www.paloaltonetworks.com

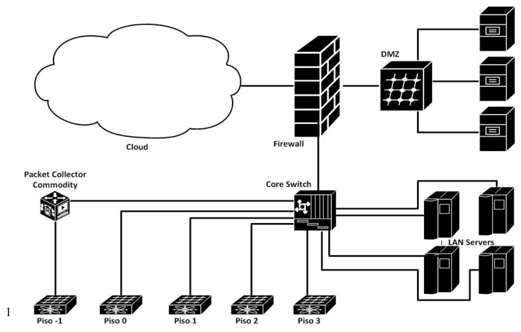

Figure 3 - Data collecting ecosystem.

Fig. 4 presents the "complex network" model obtained and Table I shows numerical characteristics of the model. Clustering coefficient defines the complex networks structure as having small-world properties. We are now studying these values, its relation to the SaaS for which we have collected data, and what they can mean in that context.

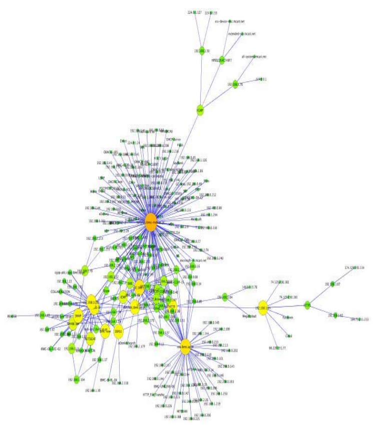

Figure 4 - Complex network structure at the SaaS level.

TABLE I. VALUES OF COMPLEX NETWORK AT THE SAAS LEVEL

Element	Value
Average degree *(k)*	2.869
Average shortest path *(l)*	3.074
Clustering *(C)*	0.004

Another experience was conducted within the ISOFIN project to assess the characteristics of an information system architecture that is being designed. The ISOFIN project [43] aims to provide a set of functionalities based on the cloud paradigm as defined by NIST [44] and enacting the coordination of independent services relying on private clouds in a coordinating public-cloud application (the ISOFIN Platform). The resulting ISOFIN platform will allow the semantic and application interoperability between enrolled financial institutions (Banks, Insurance Companies and others). In the presented real industrial case, the process-level 4SRS method [45, 46] is used to create the necessary context to elicit the requirements for designing an architecture capable to be implemented in the three typical cloud-layers: IaaS, PaaS (platform as a service), and SaaS.

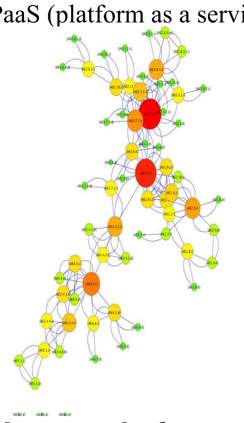

Figure 7 - Complex network of process-level logical architecture of ISOFIN in a circular layout with expansion of packages and relevant edges

Fig. 4 and Table I present the results from the IBMC experience; they allowed discovering the existence of "small-world" network properties in the existing information system. Fig. 5 and Table II present the results from the ISOFIN experience; they allowed discovering the existence of communities that are not equal to the packages defined in the designed architecture for the ISOFIN information system. A study of the construct communities should also be addressed in the future work of this PhD.

TABLE II. VALUES OF COMPLEX NETWORK AT THE SAAS LEVEL

Element	Value
Average degree (k)	2.735
Average shortest path (l)	4.150
Clustering (C)	0.253

V. FUTURE WORK AND EXPECTED RESULTS

For the next two academic years (2012/13 and 2013/2014), we will address the other two levels (users and IaaS). With the lessons learned from these three kinds of experiments we expect to define the main elements for the complex adaptive information systems architecture.

VI. CONCLUSIONS

Organizational information systems are today faced with a need for management of information flow through space and time in order to support organization information needs. Inspired on the work been done on the definition of WWW and social relations, this PhD proposal is presenting a new approach to the modeling of organizational information systems architectures, by using the "complex networks" concepts. It adopts the physics concepts of complex networks and proposes a research agenda for modeling organizational information systems architectures as a first step to the engineering of information systems. Although those concepts have already been used in modeling the WWW, power grids or air traffic systems, they have never been tested in the information systems domain. The collaboration of an enterprise like Palo Alto Networks, making available the last version of a context firewall in order to be used for data collection, makes a big impact on what can be collected and on what models that can be produced.

A special attention was paid to the choice of the selected organizations for data collection, trying to address relevant domains in order to give the vision for the fitness of the "complex networks" concept in different context and behaviors. This approach addresses the use of physics concepts that once more shows that information systems research benefit from the knowledge any area domain. Its different way of seeing things can be a central research for the leverage of organizational information systems architecture to the center of co-evolution of organization socio-technical systems trough the ability to exploit architecture for agility, flexibility and adaptability

REFERENCES

[1] Barabási, "The Architecture of Complexity," *Control Systems Magazine, IEEE,* vol. 27, pp. 33-42, 2007.

[2] S. H. Strogatz, "Exploring complex networks," *Nature,* 2001.

[3] T. H. Davenport, "The coming commoditization of processes," *Harvard Business Review,* 2005.

[4]N. Carr, "The end of corporate computing," *MIT Sloan Management Review,* vol. 46, pp. 67-73, 2005.

[5] N. Carr, "IT doesn't matter," *IEEE Engineering Management Review,* 2004.

[6] A. R. Hevner, S. T. March, and J. Park, "Design science in information systems research", *MIS Quarterly,* 2004.

[7] Y. E. Chan, "Why haven't we mastered alignment? The importance of the informal organization ," *MIS Quarterly Executive,* 2002.

[8] A. W. Don Tapscott, *Wikinomics - how mass collaboration changes everything*. New York: Portfolio, 2006.

[9] Y. E. Chan and R. Sabherwal, "Antecedents and outcomes of strategic IS alignment: An empirical investigation," *IEEE Transactions ,* 2006.

[10] K. Lyytinen and M. Newman, "Punctuated equilibrium, process models and information system development and change: towards a socio-technical process analysis," *Sprouts,,* 2006.

[11] R. J. Paul, "Challenges to information systems: time to change," *European Journal of Information Systems,* 2007.

[12] P. Beynon-Davies, "Informatics and the Inca," *International Journal of Information Management,* 2007.

[13] P. Beynon-Davies, "Neolithic informatics: The nature of information," *International Journal of Information Management,* 2009.

[14] S. Alter, "Defining information systems as work systems: implications for the IS field," *European Journal of Information Systems,* 2008.

[15] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," *Fourth International Conference on Web Information Systems Engineering (WISE'03),* 2003.

[16] E. Bonabeau and C. Meyer, "Swarm intelligence: A whole new way to think about business," *Harvard Business Review,* 2001.

[17] A. Desai, "Adaptive complex enterprises, *Communications of the ACM,* 2005.

[18] G. Hamel, "O Futuro da Gestão," 2008.

[19] I. Nonaka and G. v. Krogh, "Perspective---Tacit Knowledge and Knowledge Conversion: Controversy and Advancement in Organizational Knowledge Creation Theory," *Organization science*, 2009.

[20] J. Surowiecki, *A sabedoria das Multidões Como Inteligencia colectiva transforma a economia e a sociedade*, 1^a ed.: Lua de Papel, 2007.

[21] M. Iansiti and R. Levien, "The keystone advantage," *harvardbusiness.org,* 2004.

[22] M. Schneider and M. Somers, "Organizations as complex adaptive systems: Implications of Complexity Theory for leadership research," *The Leadership Quarterly,* pp. 351-365, 2006.

[23] J. Sutherland and W. J. van den Heuvel, "Enterprise application integration and complex adaptive systems," *portal.acm.org,* 2002.

[24] K. J. Dooley, "A Complex Adaptive Systems Model of Organization Change," *Nonlinear Dynamics, Psychology, and Life Sciences*, 1997.

[25] K. J. Dooley, T. L. Johnson, and D. H. Bush, "TQM, chaos and complexity," *Human Systems Management,* 1995.

[26] J. Holland, "Studying Complex Adaptive Systems," *Journal of Systems Science and Complexity*, 2006.

[27] J. H. Holland, "Adaptation in natural and artificial systems," *mitpress.mit.edu,* 1992.

[28] M. Mitchell, *Complexity - A guided Tour*: Oxford University Press, 2009.

[29] K. Popper, "Three worlds," *bengin.net,* 1979.

[30] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of networks," *Advances in Physics,* 2002.

[31] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of Networks - From biological nets to the internet and www," *Oxford University Press,* 2010.

[32] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, "Critical phenomena in complex networks," *Reviews of Modern Physics,* 2008.

[33] M. Newman, *Networks: An Introduction*: Oxford University Press, Inc., 2010.

[34] M. E. J. Newman, "The structure and function of complex networks," *SIAM review,* 2003.

[35] J. A. Zachman, "The Zachman Framework," *, Institute for Framework Advancement. Available ,,* 2007.

[36] J. Zachman, "Enterprise architecture and legacy systems, getting beyond the "legacy"," 2004.

[37] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal,* 1999.

[38] A. Gerber, P. Kotze, and V. d. Merwe,A, "Towards the formalisation of the TOGAF Content Metamodel using ontologies," *ICEIS,* 2010.

[39] S. Leist and G. Zellner, "Evaluation of current architecture frameworks," ACM symposium on Applied computing, Dijon, France, 2006.

[40] M. R. De Villiers, "Three approaches as pillars for interpretive Information Systems research: development research, action research and grounded theory," *SAICSIT,* 2005.

[41] M. C. Gonzalez and A.-L. Barabasi, "Complex networks: From data to models," *Nat Phys*, 2007.

[42] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *NIST*, 2009.

[43] Research Project: http://www.i2s.pt/i2ssite/Projectos/isofin.asp

[44] NIST, www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf

[45] Nuno Ferreira, Nuno Santos, Ricardo J. Machado, Dragan Gasevic. Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. PROFES, 2012.

[46] Ricardo J. Machado, João M. Fernandes. Heterogeneous Information Systems Integration: Organizations and Methodologies. Markku Oivo, Seija Komi-Sirviö (Eds.), Product Focused Software Process Improvement, LNCS Springer-Verlag.