Applied Research Processes for Software Systems Development in Industry

João Miguel Gonçalves[1][2]

[1] MAP-i, Rua Campo Alegre 1021/1055,
4169 - 007 Porto, Portugal

[2] PT Inovacao, Rua Eng. José Ferreira Pinto Basto,
3810-106 Aveiro, Portugal

joao.m.goncalves@fc.up.pt

Abstract. A research project's output is required to be useful for the organization that is hosting it. Because of this, specially in an industrial setting, IT research is usually conducted at a very experimental level, almost always involving prototype development. In this paper, the author attempts to compile relevant approaches when developing innovative software artifacts. Moreover, the author will attempt to demonstrate the strategic value of design decisions when conducting this kind of research.

Keywords: prototype, research, design, state of the art

1 Introduction

The software development industry, at least at its top level, is one of the most concerned industries regarding the innovation that their products bring forward. This happens basically because, as in the pharmaceutical industry, the cost of development is far greater than the cost of production. If properly developed and matured, software artifacts can be replicated and re-used with marginal cost. Basili [7] goes further by claiming that in software we do not re-produce the same object of software. That is why typically big software houses are very involved in the research communities and standardization bodies, and work using levels of abstraction and generally accepted software concepts. In opposition, small software companies often rely in customer-tailored solutions that provide a fast return on investment but are hardly re-usable.

This paper openly advocates the practice of implementing proof of concepts when researching in an industrial setting, and attempts to show the benefits of doing so. In chapter 2, current research and development practices are taken into account, and their academic and industrial relevance is evaluated. Then, prototypes as enablers for social related research upon the outcome of more technical research tracks are discussed in chapter 3. Furthermore, in chapter 4, the importance and role of reviews of the state of the art are addressed and finally, in chapter 5, a set of concerns that a prototype designer should have is discussed.

2 Why Develop a Prototype

While the definition of "innovative" is quite clear, the practical classification of software related research as innovative or not is prone to discussion, both in academia and in industry. Given that the definition of innovative is clear and assuming that the state of the art is known, doubts about this are expected to occur when the research results are not specific enough. Several papers address the ethereal nature of software [1][2] and mention the difficulty to represent it as rigorously as, for example, an architect represents a building by drawing sections or building maquettes. This fact is a major contributor to the referred specification problem. The problem is likely to occur if the research result is expected to originate new software artifacts, regardless if the focus of a research project is closer to computer science or to software engineering. Exceptions are non-functional innovations contained in predefined software components, that have marginal influence on the rest of the system.

One can attempt to tackle the validation of this kind of research without implementing a prototype, and effectively demonstrate the research results, but the effort to do so is, in many cases, comparable to the effort of actually implementing the concept. After all, code written in an high level programing language is only a few steps away from specification languages like VDM-SL or Alloy. While code effectively defines the software artifact, it might be argued that this kind of extreme specification is excessive and may cripple the original concept. In the other hand, this extra specification effort may help raising questions relevant for the concept that had been previously overlooked, working as a maturation method. Also there are coding practices that can help to maximize the code fidelity with respect to the original concept, like complete decoupling in the code between what belongs to the concept and what is accessory. Also, generalizing explanations can be made when necessary. It is this kind of experimental research that drives forward many other fields, based on cycles of hypothesis formulation and subsequent experimentation [7]. The development of a software system that proves that a theory holds is not only a validation of the current theory, but also a working base for further theories to be formulated.

Concept implementation is a relatively used practice in Software Engineering (SE) research, with approximately 17% of the papers using it. This number becomes even more relevant if compared with less than 3% observed in papers regarded as Computer Science (CS) research. [3] Also, using this as validation is usually appreciated by SE conference reviewers. In a study conducted by Shaw in which the submitted papers to the 2002 International Conference on Software Engineering were analyzed [6], the papers that addressed validation using an experience or example had an acceptance ratio well above average 24% for experience (the highest scoring method) and 20% for example based validation, comparing with 14% global acceptance ratio.

3 Further Use of Prototypes

Besides the research validation advantages of developing prototypes, in an industry setting the resulting prototype has added value, essentially for two reasons: it can be reused and it can be analyzed. When developing other software components, the prototypes can be used as a black-box for fulfilling a function in the system, if no suitable component exists. Another type of reuse can occur if that very research project is to be turned into a production project. The prototype can be taken as basis and the product can grow [1] from it, by incrementally replacing or improving the parts that do not meet production requirements. We would then have out of the box integration right from the start of the development, that enables early end to end testing and participatory design [2].

In this scenario, research related to the Information System (IS) body of knowledge can be conducted on the existing prototype. Namely, it is possible to start analyzing the behaviour of the prototype system. This practice fits the systems development research in IS outlined by Nunamaker and Chen in [4]. The behaviour of the system can be analyzed at a first stage by doing conceptual analysis and controlled laboratory experiments. These are two commonly applied research methods in IS, as are case and field studies [3], although these last ones should only take place at a later stage of the system's maturity.

It becomes clear that the development of a prototype during the CS or SE research process eases the transition to IS research focusing the whole system in which the original research was conducted. The IS research will then help with the system maturation process and will identify problems that relate to the usefulness and impact of the conducted CS and SE research, possibly providing strategic CS and SE research directions. The notion of the importance of this cycle has been present for a long time in the community and can be illustrated by a Hitch and McKean's statement dating from 1960 [5]: "without development, research has no use; without research, development has no base."

4 Assessing the State of the Art

A comprehensive review of the state of the art is the starting point of much research work. This is illustrated by the number of citations that state of the art surveys usually have, much higher than regular papers. In fact, it is not only common but also desirable that surveys, besides analyzing past research, also point future research tracks [8], enabling other authors to easily pick up where the survey stopped. One should not restrict itself to one survey, though. They can be thought as milestones , but a forward search, through the citing articles, must also be done. Many times it is the case that articles from similar areas that weren't included in the survey also matter to the work, or even work from different disciplines [8]. These kinds of reviews are important because they synthesize the literature. The introduced concepts by the regular research papers are linked and some concept classification or organization takes place. Ultimately, knowledge becomes structured, enabling convergence and the formulation of wider theories and paradigms.

When thinking in an applied research context, the considered survey does not limit itself to the state of the art of that research track. The system in which the track belongs to should be subject to a review. This will help to assess the development trends of the system and how the newly introduced changes should be integrated. In a more practical point of view, relevant open source projects and relevant projects from the research institution(s) and partners should be studied. This can be of value both when designing the concept implementation and for enabling synergies that increase the industrial value of the performed research work.

5 Designing Prototypes

The specificity problem discussed in chapter 2 is more likely to occur when researching about concepts that heavily interact with the systems around it. As mentioned previously, if the targeted problem is a contained one and it relates to non-functional areas, like performance or efficiency, it might be possible to abstract the impact in the rest of the system. Most of the times this is not the case, thus a concept implementation is useful.

Developing an integrated proof of concept can be a very complex design task. In this context, the design activity has special contours: the *means* for fulfilling the *task* [2] are not as readily available and the resources are typically much lower than for production projects. It is the designer's task to determine where and how effort can be saved without compromising the researched concept nor deforming its system. The task of the designer can be eased if there is a comprehensive knowledge of the *means*, i.e. a good state of the art that includes not only the research focus but the system that it integrates.

Additional design challenges can arise in collaborative research projects, where different research objectives are targeted by different people, but belong to the same system. The individual research goals may be completely diverse and sometimes may place requirements in the system that did not exist in the state of the art. It is the designer's role to harmonize the research impact in the system and to attempt to fulfill the requirements on the system without significantly increasing the required resources or time.

6 Conclusion

Arguably, the most relevant part of this paper is the identification of two distinct research cycles. One is an experimental cycle within CS and SE research, that relies in a repetition of the theorization, implementation and experimentation sequence in order to advance the state of the art. The other is a reflective/evaluative cycle more linked to the IS field that can, nevertheless, provide input that can be used by companies and institutions dedicated to CS and SE research.

The implementation of prototypes that translate as closely as possible the concept and that takes in account the system it integrates can prove to be of great value, specially in an industrial setting. For that, a comprehensive review of related work and thoughtful design work are essential. This kind of approach can enable further research work and intra or inter institutional synergies that are beneficial to the advancement of the state of the art.

References

1. Brooks Jr, F.P.: No Silver Bullet: Essence and Accidents of Software Engineering. (1986)
2. Taylor, R.N. & Hoek, A.V.D.: Software Design and Architecture The once and future focus of software engineering. Future of Software Engineering, IEEE Computer Society, pp. 226-243 (2007)
3. Glass, R.L., Ramesh, V. & Vessey, I.: An analysis of research in computing disciplines. Commun. ACM, 47(6), 89-94 (2004)
4. Nunamaker, J. & Chen, M.: Systems development in information systems research. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, pp. 631-640 vol.3 (1990)
5. Hitch, C.J. & McKean, R.N.: The Economics of Defense in the Nuclear Age. (1960)
6. Shaw M.: Writing Good Software Engineering Research Papers. Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, pp. 726-736 (2003)
7. Basili, V.R.: The Role of Experimentation in Software Engineering: Past, Current, and Future. Proceedings of the 18th International Conference on Software Engineering, IEEE Computer Society, pp. 442-449 (1996)
8. Webster, J. & Watson, R.T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review. Management Information Systems Quarterly, 26 (2002)