Software Engineering in Practice: Building Software Factories

Jones Albuquerque, DFM - UFRPE. Recife, Brazil. {joa@ufrpe.br}
Silvio Meira, CIn - UFPE / CESAR. Recife, Brazil {silvio@cesar.org.br}

Abstract

Practical issues of real software development have not been considered in Software Engineering Graduate Programs. Traditionally, the programs only present to their students some new technology or recently scientific aspect but do not present real aspects in practice of software development. This position paper presents an experimental initiative on incorporate practical issues of software development using Software Factory concepts in a graduate course. Real Software Factories, with real projects, with real deliverables are setting up in four months. Traditionally, the term Software Factory has the erroneos connotation that software development is comparable to mass-production of industrial products, and this is not the case. We present a proposal to build real Software Factories and present and discuss its results.

Keywords: Education, Software Engineering, Development Process, Software Factories.

1 Introduction

The term “Software Engineering” was originated in 1965 but first come into currency in 1967 when study group on Computer Science of the NATO Science Committee called for an international conference on the subject. As Brian Randell and Peter Naur point out in the introduction to their edition of the proceedings, "The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be [based] on the types of theoretical foundations and practical disciplines[,] that are traditional in the established branches of engineering.” [3]. This setence opens several areas of potential disagreement. Just what are the "types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering”? What would their counterparts look like for software engineering? What role does engineering play in manufacture? Could one assign such a role to software engineering? Can software be manufactured?

That questions had no definitive answers in the conference proceeedings and among the future Software Engineers. If one could not define “Software Engineering”, how could one point to its practice? In 1971, at IFIP, F. L. Bauer put in his report on “Software Engineering”: *What have been the complaints? Typically, they were:*

1. *Existing software production is done by amateurs (regardless whether at universities, software houses or manufacturers);*
2. *Existing software development is done by tinkering (at the universities) or by the human wave ("million monkey") approach at the manufacturer's;*
3. *Existing software is unreliable and needs permanent "maintenance", the word maintenance being misused to denote fallacies which are expected from the very beginning by the producer;*
4. *Existing software is messy, lacks transparency, prevents improvement or building on (or at least requires too high a price to be paid for this);*
5. *Last, but not least, the common complaint is: Existing software comes too late and at higher costs than expected, and does not fulfill the promises made for it.*

In this way, Bauer observed that “Software Engineering” seems to be well understood. Traditionally, these had been the concern of engineers, rather than of scientists [1]. Nowadays, these complaints are yet present in Software Industry "reports" and Academic "papers" [2].

2 Practical Results: Building Software Factories

Building Software Factories courses expose students to real, team-oriented development in a software development organization staffed and managed by students under the guidance of faculty. Several students are professional developers, certified programmers and work in industry, too. These courses are hands-on courses that require student participation in one of software factories defined. The class meet once each week. One class meeting lasts two hours and is usually led by the professor.

During these meetings, the professor introduces concepts that are relevant to the current work being performed in the factories and addresses problems faced by the students at the factories. The professor is a "facilitator" who does not decide right or wrong, but instead facilitates learning the pitfalls and peaks in development process. This is not an innovative initiative as presented in [5]. The innovative aspect is in the time that the factories are build: four months!

The projects for each software factory is chosen by professors and software factory managers. The demand are caracterized by RFP - *Request For Proposals* and have one client per project. These projects are in collaboration with CESAR (HTTP://WWW.CESAR.ORG.BR) which reflect current trends in industry and makes its professionals (which are students in the course) motivated [6].

The lastest course edition can be found at HTTP://WWW.CIN.UFPE.BR/~IN953/. In this site there are papers, slides, RFP's, software factories sites, and experience papers published by students relating their experiences in the course. A typical calendar to build the factories is:

1. Concepts and definition - 1 month;
2. RFP to calibrate the factories - 1 month;
3. Real RFP to evaluate the factories - 2 months.

The three lastest software factories which are hybrid-open source software communities [4] have their sites published at:

- OpenGadgets
  HTTP://WWW.CIN.UFPE.BR/~OPENGADGETS
- Usina
  HTTP://USINA.TIGRIS.ORG/
- Engenho de Software
  HTTP://WWW.CIN.UFPE.BR/~ENGENHO/

3 Conclusions

In recent years, there have been a number of experiences with software factories reported in the literature which, although dense in a number of ways, have been lacking in what regards the discussion of the stages of definition and setting up of factories themselves.

This work discusses a number of issues related to the conception, implementation and improvement of real software factories and, as a result of a real life experiment, also points to a number of lessons learned, which can very likely be replicated within similar contexts.

References

[1] I. Aaen, P. Bøtcher, and L. Mathiassen. Software factories: Contributions and illusions. In *Twentieth Information Systems Research Seminar*, Scandinavia, Oslo, 1997.

[2] J. H. Johnson. Micro projects cause constant change. Technical report, The Standish Group International Inc, 2001. CHAOS Report.

[3] M. S. Mahoney. The roots of software engineering. Technical Report CWI Quarterly 3-4, Princeton University, 1990. pp 325-334.

[4] S. Sharma, V. Sugumaran, and B. Rajagopalan. A framework for creating hybrid-open source software communities. *Info Systems*, 12:7–25, 2002.

[5] J. D. Tvedt, R. Tesoriero, and K. A. Gary. The software factory: Combining undergraduate computer science and software engineering education. *IEEE*, 2001.

[6] K. Wiegers. Creating a software engineering culture. In *Software Development*. Process Impact, July 1994.