Software evolution the need for empirical evidence

[Position Paper]

Giuliano Antoniol, Yann-Gaël Guéhéneuc, Ettore Merlo, and Houari Sahraoui
Ecole Polytechnique and University of Montreal
Montreal, Quebec, Canada
antoniol@ieee.org, ettore.merlo@polymtl.ca, {guehene,sahraouh}@iro.umontreal.ca

An intrinsic property of software is its malleability, the fact that it may change and evolve. Software evolution is costly, because software systems tend to be highly complex and large. They are highly human-intensive and risky, because unplanned and undisciplined changes in any software system of realistic size risk degrading software quality and may produce unwanted and unexpected side effects. As a software system is enhanced, modified, and adapted to new requirements, its code becomes increasingly complex, often drifting away from its original design. The current state-of-the-art in software evolution offers only short-term solutions to software change and evolution focused on software maintenance and defect repair, in which only the source code evolves, while the architecture, design, and—more generally—the documentation are not updated.

The recent change towards a global economy poses new challenges with respect to software evolution. Software systems represent valuable assets to be preserved over time. Change and evolution management policies and strategies involve at least two different levels of granularity:

- First, the **macro level** at which a company manages its IT portfolio and makes macro-economic decisions. For instance, should a company wish to replace an existing IT component with a new software system, it must first conduct a feasibility assessment. The variables involved include the company's financial status, anticipated market trends, the costs of acquiring a new software system and training employees, total cost of ownership of the new software, and the cost of data migration. Other variables driving the decision have to be considered as well: the risks associated with possible failure in data migration or with the software vendor's financial status and liability.
- Second, the **micro level** refers to project-level company decisions. Once the decision to evolve the proprietary software has been made, the software change and evolution process must be implemented in facts. Implementation in turn calls for decisions such as partially or entirely outsourcing the maintenance and evolution process, defining the staffing level and evolution plan, reshaping the workforce, organizing the work packages, and defining schedules.

The already complex situation is made even more challenging by a new industry trend. Traditionally, the choice was either to *make* or to *buy*. However, the new market created by the service-oriented paradigm offers a new opportunity: Instead of buying or making, it is now possible to *rent* a service.

To make the situation even more complex, the cost of labor in emerging[1] countries favor outsourcing and distributed software evolution. For example, in 2003, the Russian offshore programming industry earned total revenues of 546 millions $US, according to figures compiled by CNews Analytics and Fort-Ross, an association of Russian software companies. The same report projected growth rates of 30% to 40% for the coming few years, meaning that the industry could cross the 1 billion $US by 2006. Overall, according to the analysis firm Datamonitor, the IT outsourcing market grew by 37% to 163 billions $US in 2004.

Distribution enables hiring the best experts without regards for geographical distances, thus reducing costs. Outsourcing rates tend to vary by the strength of the home economy and its degree of maturity of its IT industry. The cheapest alternative today is China, followed by Eastern European countries. India has a more mature IT industry, which makes outsourcing there more costly. However, although distribution allows emerging countries to attract investments, it also tends to draw resources and job opportunities from developed economies to relocation in third world countries.

[1] By "emerging countries", we mean countries in which the IT industry is becoming mature and a major source of incomes for the countries.

As a research community, we share the responsibility to support economic growth, employment, and training of highly qualified personnel. Thus, any action is important that may lead to identify approaches, methodologies, and tools, capable of enriching our current body of knowledge and of contributing to understanding the mechanisms in distributed software evolution. Indeed, labor cost and conditions are a major social issue worldwide. We are responsible of the effects and influences of our research results on these labor cost and conditions. In particular, we are responsible for devising approaches, methodologies, and tools to improve the general economy worldwide, such that outsourcing is beneficial to all, considering the difference of labor cost between developed and emerging countries (about an order of magnitude). This is a matter of scientific ethics.

Distributed software evolution needs convincing evidences to support the adoption of cost-effective and ethical solutions, as well as solutions that prove to be beneficial worldwide in the long term. For example, regardless of the adopted paradigm, technology, tools, there exists a very limited body of empirical results to help deciding what part of the software evolution process should be outsourced, where and when it should be outsourced, and what are the short-, mid-, and long-term effects of such a decision. Other engineering disciplines developed throughout time a body of knowledge to choose methods and to predict results effectively. By mean of standardization, subcontract standardization, and the adoption of components, industry in general increases productivity and decreases costs substantially. In software engineering, we are still far from the maturity of other engineering fields. Researchers and practitioners struggle to provide guidelines, evidences, and to meet predicted costs, schedules, and quality levels. Clearly, there are commonalities and differences with general industry, and general industry approaches may not apply in software engineering. (In particular, software is immaterial: There is easy means neither to verify where it was developed, maintained, evolved, nor to verify if it complies with standards, regulations, labor conditions.) However, as a research community, we must strive in developing approaches, methodologies, and tools, to meet practitioners' and societies' need alike.

In field validation, blind and double blind studies are mandatory to pharmaceutical industry prior a drug enter the market; in the same area replicated studies and studies disproving prior beliefs are as relevant as studies suggesting a beneficial application of a certain drug to a given disease. Standard ways to design and to carry out drug experimentations have been defined and are used. A solid scientific basis and legislation attempt to prevent misconduct experiments, flaw experiment design, wrong assumption and conclusion. We believe that in the software engineering area much as for pharmaceutical companies there is a need for a more solid scientific basis, a standardized way to perform experiments, more replicated studies, and rules for accepting replication studies in conferences and journals.

We identify several research questions that ought to be address by our research community regarding distributed software evolution:

- How could levels of productivity be increased? What approaches, methodologies, and tools use?
- Which parts of the distributed software evolution can be de-localized? What are the conditions to verify, the constraints to check, the controls to perform?

More generally, we believe that our research community must strive to answer the following questions:

- Is maintenance a fatality? How is maintenance performed? How is maintenance validated?
- How to compare the results of empirical software engineering studies? What approaches, methodologies, and tools should be applied?
- How to validate pieces of software (as for drugs)? Is it possible, desirable?
- How to perform and to publish replication studies?