Requiem for software evolution research: a few steps toward the creative age

G. Antoniol
Département de Génie Informatique, École Polytechnique de Montréal
C.P. 6079, succ. Centre-ville Montréal (Québec) H3C 3A7 Canada
antoniol@ieee.org

ABSTRACT

Nowadays almost every company depends on software technologies to function, the challenge is that the technologies and software applications are constantly changing and adapting to the needs of users. This process of change is risky, since unplanned and undisciplined changes in any software system of realistic size risk degrading the quality of the software and producing unexpected side effects. The need for disciplined, intelligent, cost-effective software change and evolution is an urgent technological challenge in the software engineering field.

New technologies, new social and cultural trends, a widespread adoption of open source software, the market globalization and new development environments are spelling the requiem to the traditional way in which software evolution research was carried out. Evolution research must evolve and adapt to the new society needs and trends thus turning challenges into opportunities. This keynote attempts to shed some light on key factors such new technology transfer opportunity, the need of benchmarks and the three items each and every research program in software evolution should integrate in one way or the other.

Categories and Subject Descriptors

D [**Software**]: Miscellaneous; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement

General Terms

Software Evolution, Technology Transfer, Innovation

Keywords

Software evolution, Open source, IT trends

An intrinsic property of software is its malleability, the fact that it may change and evolve. Changes and evolution characterize the entire life span of any software system, from inception to retirement. The need for disciplined, intelligent, cost-effective software change and evolution is an urgent technological challenge in the software engineering field. Successful software systems operate for decades, and often outlive the hardware and operational environments for which they were originally conceived, designed and developed. In real-world environments, software must constantly evolve, or it is doomed to become obsolete. New technologies, new players, new cultural and society values and a constant strive to reduce software total cost of ownership are changing the software landscape and challenging the software evolution community. Key factors such as:

- Service Oriented Architectures (SOA);
- Autonomic, self configuring, self healing systems;
- Market globalization, new social and cultural trends toward a creative society;
- Widespread adoption of open source software; and
- New powerful development environments.

Are influencing the way in which software is developed, evolved and eventually retired. Indeed, the choice is no longer limited to make or buy; SOA and the cultural shift to view software as providing a service [6, 3] already makes it possible to rent a software or the service implemented on top of it.

Despite the profound changes we observe, the current state-of-the-art offers only short- term solutions focused on the mere technical issues of software maintenance and defect repair. Software development and evolution are social phenomena involving technical as well as non technical matters but only recently social networks, cultural issues and financial aspects have drawn the attention of researchers, see for example [8]. Even worse, we are lacking of theories, methodologies and approaches providing systematic support for disciplined, long-term change planning. Software system change and evolution is costly, since real world software systems tend to be highly complex and large in size. It is highly human-intensive and risky, since unplanned and undisciplined changes in any software system of realistic size risk degrading software quality, and may produce unwanted or unexpected side effects. In addition to these technical difficulties, it is also impossible to predict how a software system should function in the future, which means that available software design techniques cannot help to fully anticipate the long-term evolution of a software system. Technical challenges are not the sole problem. Outsourcing in emerging countries reduces costs but it also imply that applications are often evolved by non native English speakers. However, software manuals, design documents, requirements are likely, if they exist, to be written in English. Calling service centers of software and hardware vendors may turn into an amazing social experience where non native English speakers attempt to communicate in a language they believe is English.

According to recent reports produced by Forrester Research[1] [2], despite concerns on copyright and patent infringement, open source adoption for mission-critical applications is no longer a taboo. About 80 percent of respondents report using open source in the application infrastructure providing the underpinning for both routine and mission-critical applications. Such a widespread infrastructural utilization indicates a strong potential and relevance of research results obtained by studying open source infrastructural components such as databases, Web servers and application servers. The relevance of open source case studies has been anticipated by the maintenance and evolution community. The workshop on mining software repository and an increasing number of papers reporting empirical evidence related to open source projects are encouraging facts. We must recognize that Apache [4] and Firefox [9] account for about 50 % and 15 % of market share respectively; Eclipse exceeds 65 [5] percent of Java IDE market share. In other word, the mantra "you have not industrial data" have to be reformulated in "you have not realistic data"; research and studies relying on large open source software are no longer second rank contribution. Eclipse, is giving to the software evolution community a tremendous opportunity setting the stage for profound revolution. Technology transfer from research to market depends on several factors such as the domain e.g., automotive versus information technologies, the regulations e.g., food and drug administration, the market opportunity e.g., lacking of competitive technologies. Eclipse integrates software evolution research ideas dating back to early nineties e.g., automatic refactoring; its architecture of open framework make it easy to implement and deploy new plugins. By developing our plugins and making them available over the network the technology transfer can be substantially shortened and likely measured in years instead of decades.

According to IDC and The Economist the market of software and software services will still increase far more then the hardware market. Services are ubiquitous and will reach almost 100 % of company adoption in the next couple years to come. A global market makes it easy to find early adopters for disruptive technologies and our community has to take the challenge and turn it into a competitive advantage. No matter the platform traditional applications were developed for, applications need to migrate to Web services, Web services need to be evolved, business intelligence and autonomic computing needs to be integrated into systems. The autonomic computing initiative launched by IBM cannot be overlooked. Researchers in software evolution need to recognize that evolving a single, isolated, small non distributed application running on a single host is easy. The challenge is when we have to evolve a distributed application running on a distributed server farm where tens of thousands of machine provide critical services to our society.

A shift in labor market and a worldwide economy promotes a rise in creative jobs [1] such as computer, mathematical and engineering occupations. Researchers and practitioners are moving across the borders following criteria no longer limited to the salary. The 2001 information week survey reports salary as only the forth motivation to chose a job; 67 % of respondents rank "challenges" as the top motivation [1]. The software evolution community has to recognize the rise in creative contributions from regions once only marginally considered. China, India and Western Europe are rapidly becoming major research players substantially contributing to technology and innovation. These countries have in place high quality education programs; the China government launched major initiatives in high technology training. The pressure to increase productivity and decrease software total cost of ownership already forced Indian companies to outsource activity to China. The same pressure will bring into play researchers and facilities of those countries. The share mass of intellectual power coming into play challenge any traditional academic research approach; research groups and communities not focusing on key and challenging research problems not measuring, quantifying and assessing results on sizable test beds are domed to find difficulty to attract new blood and likely to disappear.

[1]http://www.mywebservices.ca/news-im/07-312-may-16.html

The software evolution community should address the difficulties faced when striving to make software change and evolution more cost-effective. Software changes should be carefully managed. More specifically, software evolution researchers should develop sophisticated tools for the effective planning, managing and implementation of software changes for sizable applications and systems. These tools should be able very easy to use, they should pass an equivalent for computer science of the "mom test" [7]; if my mom, nowadays 75 years old is not adopting a technology you are loosing an important market share and your technology is likely not to be a mature technology. Much in a similar way, if programmers and managers are not adopting what we are developing in our laboratories then we have a problem. Indeed, one of the key indicators of a mature technology is the ease of use for both non-experts and professionals. Any research agenda in software evolution should consider integrating in one way or the other, in total or in part, some aspects of the following **key objectives**:

- Managing software evolution projects: to identify models for planning and managing software change and evolution in a cost-effective way;
- Understanding software evolution: to identify architectural and design solutions promoting software evolution; and
- Retrospective analysis of evolving software: to investigate and analyze information from previous software releases and versions in order to improve future versions.

A further challenge for our community is the lacking of benchmarks, of a common yardstick. Despite an encouraging trend toward the use of applications such as Mozilla, Eclipse or Jboss as test beds there exists a very limited body of empirical results. Empirical investigation into the cost and effectiveness of change and evolution strategies should be a major component of any research program, however, few case studies and simulations have been performed to understand what makes a software system able to evolve, to identify optimal or sub-optimal change schedules and change strategies, and, more generally, to validate approaches that will reduce evolution costs. Empirical studies are needed to demonstrate the effectiveness of our research and facilitate collaborative and evolutionary work among researchers and practitioners. However, there is a lack of common experimental design, methodologies, and benchmarks. Hence, work and resources are needed to define standard processes and related procedures and to build benchmarks for performing empirical studies during evolution research.

1. REFERENCES

[1] R. Florida. *The rise of the creative class*. Basic Books, New York, 2002.

[2] G. Giera. Unisys pioneers the right model for open source services. http://www.forrester.com/Research/Document/Excerpt/0,7211,39739,00.html, July 2006.

[3] W. Heuvel. *Aligning modern business processes and legacy systems*. MIT Press, 2007.

[4] Netcraft. June 2007 web server survey. http://news.netcraft.com/archives/web_server_survey.html.

[5] I. Skerrett. Eclipse gains market share in 2005. http://ianskerrett.blogspot.com/2006/03/eclipse-gains-market-share-in-2005.html.

[6] M. Turner, D. Budgen, and P. Brereton. Turning software into a service. *IEEE Computer*, 36(10):38–44, 2003.

[7] Various. Make it simple, a survey on information technology. *The Economist, Special Report on Science and Technology*, pages 1–14, October 2004.

[8] C. Verhoef. Quantifying the value of it-investments. *Sci. Comput. Program.*, 56(3):315–342, 2005.

[9] Wikipedia. Usage share of web browsers. http://en.wikipedia.org/wiki/Usage_share_of_web_browsers, 2007.