

Dynamically Evolving Requirements

John Mylopoulos and Neil Ernst
University of Toronto/Trento

It has been known for decades that changing requirements constitute one of the greatest risks for large software development projects [Lubars]. That risk manifests itself routinely in statistics that measure the frequency of failure and under-performance for such projects. “Changing requirements” usually refers to the phenomenon where stakeholders keep changing their minds on what they want out of a project, and where their priorities lie. Little attention has been paid to requirements changes occurring after a system is in operation¹, as a result of changing technologies and/or business needs. We refer to these as “dynamically evolving requirements”, to distinguish them from their design-time counterparts.

Evolving software is a fact of life, especially so in a fast-changing world. There has been considerable research on software evolution, focusing on code reengineering and migration, architectural evolution, software refactoring, data migration and integration. However, the problem of dynamically evolving *requirements* (as opposed to architecture, design and/or code) hasn’t made it yet into research agendas (see, for example, the topics listed as relevant to a forthcoming workshop on “Dynamic Software Evolution”, <http://se.inf.ethz.ch/~moriol/DSE/About.html>).

Addressing the problem will require research along several different strands. We list here four:

Understanding root causes. We are interested here in characterizing generic root causes for dynamically evolving requirements. For example, businesses are moving into network-based business models, such as service value networks and ecosystems; such trends are bound to generate a host of new requirements on operational systems that will have to be addressed by requirements engineers and software reengineers. As another example, Governments around the world have been introducing legislation to address growing concerns for security, privacy, governance and safety. This makes regulatory compliance another major cause for requirements change. The introduction of a single Act in the US (Sarbanes-Oxley Act) in 2002 resulted in a monumental amount of change for business processes as well as software in business organizations. The costs of this change have been estimated at US\$5.8B for one year alone (2005).

The idea for this research strand is then to identify and characterize generic causes, so that they can be effectively addressed.

From root causes to requirements. Given a root cause (such as change of business model, or new regulation), there are many ways to transform it into a collection of new requirements for a given legacy system. We expect that requirements will be conceptualized here as new/revised goals. Conceptualization of root causes is an open question. See work by Anton et al, for example, on conceptualizing regulations in terms of rights and obligations [Anton]

Evolution mechanisms. Once we have identified what are the changes to requirements, we need to implement them by changing the system-at-hand. This may be done manually, possibly with tool support, by developing novel reengineering techniques. More interestingly, evolution may be done automatically by using mechanisms, inspired by different disciplines (biology, control theory, economics, ...). Doing research along this strand will require much experimentation to evaluate the effectiveness of different evolution techniques.

Design for evolution. Some designs are better suited for evolution than others. For example, a design that can deliver a given functionality in many different ways is better than one that delivers it in a single way. As another example, modularization has proven a great principle for evolution. What are other principles, and how can they be accommodated in routine ways during design?

¹ ... with the notable exception of research on traceability mechanisms. Of course, traceability is useful for dynamically evolving requirements, but doesn’t actually solve the problem.

An obvious implication of these research topics is that the research to be conducted will have to be interdisciplinary. Researchers from Management, Organizational Theory and Law will have to be part of the community that addresses root cause characterization and the derivation of requirements from root cause changes. Evolution mechanisms and theories that account for them have been developed in Biology, Engineering, Organizational Theory and Artificial Intelligence. Designing for evolution is a major theme for researchers working on Multi-Agent Systems, Autonomic Computing and Adaptive Software Systems.

There are deeper research issues where advances will have a major influence on solutions for the problem-at-hand. We mention three such issues:

Science of design. Adopting Herbert Simon's vision [Simon], we envision here a theory of design that encompasses at least three ingredients: (a) the purpose of an artifact, (b) the space of alternative designs, (c) the criteria for evaluating alternatives. Design artifacts that come with these ingredients will obviously be easier to evolve.

Model evolution. Models will be an important (perhaps *the*) vehicle for dealing with dynamically evolving requirements. Unfortunately, the state-of-the-art in modeling is such that models become obsolete very quickly, as their subject matter evolves. In Physics and other sciences models of physical phenomena do not need to evolve because they capture invariants (immutable laws).

We either need here a different level of abstraction for modeling worlds of interest to design (usually technical, social and intentional), so that they capture invariants of the subject matter. Alternatively, we need techniques and infrastructures for model evolution as their subject matter changes.

Evolutionary design². Extrapolating from Darwin's theory of evolution where design happens with no designer [Dennett], we could think of mechanisms through which software evolves without any master purpose or master designer. In a way, this is already happening with open source software. It would be fruitful to understand better open source software development as an evolutionary process and invent other mechanisms for software evolution that do not involve a master designer (aka intelligent design in some places ...) This is in sharp contrast to Simon's vision. At the same time, this is an equally compelling one.

We are optimists, so we end this position paper on an optimistic note. A precondition for any comprehensive solution to the problem of dynamically evolving requirements is that design-time requirements are properly captured and made available at run-time, in other words they are part of the running system. Accordingly, we (optimistically) predict that the days of lip service to requirements are coming to an end, as Software Engineering Research and Practice opt for lasting technical solutions in a volatile world. Growing interest in topics such as autonomic software, semantic web services, multi-agent and/or adaptive software give some evidence that this optimism is not totally unwarranted.

References

[Anton] Breaux, T., and Antón, A., "Analyzing Goal Semantics for Rights, Permissions, and Obligations", 13th IEEE International Requirements Engineering Conference (RE'05), Paris, France, 177-186, 29 August - 2 September 2005.

[Dennett] Dennett, D., *Darwin's Dangerous Idea*, 1995.

[Lubars] Lubars, M., Potts, C., Richter, C., "A Review of the State-of-Practice in Requirements Modelling", First IEEE International Symposium on Requirements Engineering, San Jose, January 1993.

[Simon] Simon, H., *The Sciences of the Artificial*, The MIT Press, 1969.

² ... or, "Darwin's dangerous idea" [Dennett].