The Darwin Project: Evolvability of Software-Intensive Systems

Piërre van de Laar,
Sjir van Loo,
Gerrit Muller,
Teade Punter,
David Watts.
Embedded Systems Institute
Eindhoven
The Netherlands
pierre.van.de.laar@esi.nl
sjir.van.loo@esi.nl
gerrit.muller@esi.nl
teade.punter@esi.nl
david.watts@esi.nl

Pierre America,
Philips Research
Eindhoven
The Netherlands
pierre.america@philips.com

Joland Rutgers,
Philips Medical Systems
Best
The Netherlands
joland.rutgers@philips.com

Abstract

This paper presents the Darwin[1] project. This applied research project is currently conducted at Philips Medical Systems and focuses on the evolvability of software-intensive systems, with as use case Magnetic Resonance Imaging (MRI) systems. We not only discuss evolvability of software-intensive systems in general, but also describe the project and its research areas.

1. Introduction

The world we live in is constantly changing. These changes impact the software-intensive systems that are currently being used and developed. In particular, we see the following trends:

- The number of systems released per year increases. More variety is needed since the expectations of a system increasingly become tailor made: the system must be fit for purpose. It must fit the specific goal at a specific time of the specific market segment, or even organization or person. When systems become fashionable, as happened with MP3 players and mobile phones, monthly instead of yearly releases were needed to be able to follow or create the latest hypes and trends.
- The development of a system is becoming more and more global. In addition, more people and companies are becoming involved. In the past, systems were developed by a single company in a particular country. Currently, systems are typically developed by multiple companies around the globe. Baldwin and Clark accurately discuss this trend in the PC industry [3]. Furthermore, the area for which a system is developed grows. In the past, systems were developed for and sold in a particular country or region. The trend is for systems to be developed as world-products.
- The number of features increases strongly. The mobile phone is probably the best-known example: besides making phone calls, it can also send text messages, take pictures and videos, play music, browse the internet, and manage an agenda.
- Systems become part of an ever-increasing network. For example, Magnetic Resonance Imaging (MRI) systems have evolved from stand-alone scanners to a node in the hospital network.

[1] This work has been carried out as a part of the DARWIN project at Philips Medical Systems under the responsibility of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

In response to these trends, we observe that many companies invest in standardization and in open, configurable platforms (also called product lines or families) [12, 14]. Compared to a single system, these platforms have more variety, including their dynamic behavior; more complexity; more uncertainty; more features and feature interactions; more software; and require a larger team and higher integration effort. The investments in standardization and platforms are tremendous, and can only be earned back over a large period of time. Therefore, the adoption of standards and platforms enforces an incremental way of working onto a company. Evolvability is thus a serious issue for software-intensive systems.

The outline of this paper is as follows. In Section 2, we describe the Darwin project. In Section 3, we discuss evolvability in software-intensive systems in general. The research areas of Darwin with the specific focus on MRI systems are detailed in Section 4. We end with a summary.

2 The Darwin Project

Darwin is a collaborative project between the Embedded Systems Institute, Philips Medical Systems, Philips Research, and five Dutch universities (Delft, Eindhoven, Groningen, Twente, and the Free University of Amsterdam). The project started end of 2005 and will run until the end of 2010. The size of the staff of the project is equivalent to 20 full-time people, and includes 11 PhD students. The goal of Darwin is to understand evolvability as a system property; to identify, create, and apply constructs, models, and methods to support evolvability; to support the trade-off decisions the architect will have to make with respect to evolvability; and to support the sub-system and technology lifecycle view of a system. The Darwin project is executed using the industry-as-laboratory paradigm. Hence, the researchers are working closely together with developers of Philips Medical Systems. Furthermore, they have access to a large source of information, including a large code archive going back for many years.

3. Evolvability of software-intensive systems

Software-intensive systems are systems with significant mechanical, optical, electrical, and other components, but where software accounts for a large part of the value and the development effort. This is not only so-called embedded software, which somehow runs inside the system itself, but increasingly also software running on computers outside the system, which provide important services in conjunction with the system. In the Darwin project the MRI system is chosen as a use case of a software-intensive system, amongst others, since it displays virtually all of the typical evolvability issues.

For successful software-intensive systems, requirements *will* change. The change in requirements originates from the following three areas (see also [11]): First, the stakeholders' needs change. For example, with MRI systems, clinical users want to make a more accurate diagnosis and perform scans faster. Second, the operating environment of the system changes. For example, MRI systems are increasingly connected to hospital networks, and through these to the Internet. Third, the available technology for the system changes. Technology becomes obsolete [7], like windows XP for PC-based systems, and new technology becomes available [4], like Liquid Crystal Displays for televisions and computer monitors.

Some changes in the requirements can be anticipated and a system can be prepared for them. Yet, unanticipated changes will also occur since predicting the future and the consequences for a system is not an exact science (see also [8]).

Although requirements can change at any time, a system often doesn't have to change at exactly that point in time. We distinguish between three different points in time when evolution can occur: design-time, maintenance-time, and run-time. We will illustrate these three points in time using some industrial contexts.

- From ‘throw-away’ systems, like mobile phones and televisions, customers accept that they have to buy a new system whenever the gap between their current system and current requirements becomes too large. Yet, cost-effectiveness requires the producers of these systems to evolve the design over time.
- From professional systems, like an MRI system, stakeholders expect upgrades to be applied during maintenance, when it is acceptable to pause or shut down the system temporarily.
- From infrastructural and safety-related systems, like telephone switches and nuclear power plant controllers, stakeholders expect continuous availability and adaptation to changing requirements. Hence, these systems must evolve at run-time.

Depending on the decomposition of the system [10] a change in the requirement can crosscut the whole

system. But even when the change in the requirement seems to be localized in a single module, dependencies can cause the change to propagate throughout the whole system. For example, putting a more powerful engine in a car, while maintaining the same level of safety, can affect the transmission, the tires, the cooling, the brakes, and even the lights, since more power must be handled and a higher speed will be achieved while the reaction time of the driver remains the same. Change impact analysis, see for example [2], is a well-known approach to determine the total impact of a change.

Stakeholders buy a system to perform a particular function for them. They value the system based on its function and not on its constituting parts. The value of most software-intensive systems is rather constant over time. Yet, the amount of software in software-intensive systems is following Moore's law. Hence, producers of software-intensive systems are looking for methods to prevent the costs and risks associated with adopting their system to new requirements to increase over time despite the exponential increase in the amount of software.

4. Areas of research in Darwin

The business, architecture, processes, and organization (BAPO) [13] influence the system and its evolvability. We are aware that when the system must be changed [6], its evolvability is also highly dependent on the appropriateness of the processes and organization. Still, in the Darwin project we focus on architecture and consider business, processes, and organization as a given context.

In the Darwin project, we focus on three research areas: system engineering; modeling and analysis; and connecting system and software. These areas will be discussed in the following subsections.

4.1 System Engineering

This research area focuses on system engineering within a dynamic and changing environment. A system can react to changes or it can be prepared for changes. To prepare a system for a change, it must effectively deal with the uncertainty in future requirements. Scenarios, set-based design, and road mapping of markets, technology, and legislation are well-known approaches to address this topic. However, prioritizing the different changes, and deciding whether to react or to be proactive towards a particular change are (often) outside these approaches.

We would like to make well-informed architectural decisions based on cost-benefit modeling. In the past we have used several kinds of scenarios, together with adequate cost-benefit analysis techniques, to deal with uncertainty about the future [1]. Now we are developing an approach that organizes these scenarios in tree structures and uses rigorous financial instruments, such as real options theory [5], to asses the value of each scenario. In this way we expect to get a reliable estimate of the economic value of an evolvable architecture. Currently we are evaluating this approach on practical cases within Philips Medical Systems.

4.2 Modeling and analysis

This research area focuses on modeling and analysis of systems and their dynamics. Software-intensive systems contain an enormous amount of details, not only in the software, which consists of millions of lines of code, but also in the mechanical, optical, electrical, or other components. Probably, the amount of details in such a system is beyond the grasp of any human being. Yet, we want to design a system with the desired system properties. Therefore, we must abstract from the details using, for example, system architecture, interfaces, and models. In the Darwin project, we model different parts of an MRI system with the goal to combine them to predict the impact of changes on system properties like reliability, performance, and throughput. In this research area, we concentrate on three research topics.

The first research topic in this area concentrates on the communication in an MRI system: from the collection of the raw image data to the storage in a Picture Archiving and Communications System, from controlling the system to visualizing images, and from cables to communication protocols. The communication infrastructure of an MRI system impacts many system properties. For example, the amount of physical cables in an MRI system impacts the cost and ease-of-installation; the amount of connectors for the cables influences the reliability of the system; and the transport of data through the cables generates electric magnetic fields that negatively influence the image quality. A considerable part of the communication infrastructure of a Philips MRI system has been developed in-house. However, currently publicly-available and mass-marketed wired and wireless communication technologies might make it possible to replace proprietary solutions by general solutions and/or to reduce the number of physical cables by mapping multiple communication pathways onto a single cable. Models of the available hardware

and communication standards, together with a mapping from communication paths onto physical cables will be used to answer this question by determining whether the latency and throughput requirements of the whole system can be realized by publicly available technology. This research topic provides short term benefits to Philips MRI systems; it improves our understanding of the impact of disruptive technologies on the evolution of a software-intensive system; and it helps to achieve our long-term goal: designing evolvable systems.

The second research topic concentrates on the question: does the evolvability of a system increase by moving to a model-driven architecture? To answer this question, we focus on an MRI subsystem: the patient support table. This subsystem is difficult to implement due to the large combination of states and a safety requirement that ensures that the patient can always manually be moved outside the MRI system. Furthermore, this subsystem turned out to require considerable effort to evolve towards the open MRI system. In an open MRI system, patients can not only be moved in and out of the system, but also be moved sideways to position the area of interest inside the sweet spot of the MRI system. Initially, the patient support table for the closed MRI system is modeled. As expected, the challenges included the prevention of state explosion and the modeling of the safety requirement. From this model, conformance with the safety requirement can be proven, and source code can be generated. The next step is to evolve this model towards the open MRI system. Based on the experience obtained in the execution of this step, we expect to be able to tell whether this model-driven architecture was better evolvable towards an open MRI system than the original approach.

The third research topic concentrates on the change of architectural decisions, where the atomicity of MRI scans is taken as a use case. When a patient undergoes an MRI exam, multiple scans are taken. During a scan energy is dissipated into the patient and into the different amplifiers that manipulate the electromagnetic field inside the MRI system. Note that the amount of dissipated energy depends on the particular kind of scan. The dissipating energy must remain below a safety limit to prevent overheating of either the patient or the amplifiers (or both). Under the current architecture, scans are executed sequentially, and the safety limit is realized on a per-scan basis. As a consequence, exams with safety critical scans often contain considerable idle time. This raised the question: can exams be executed faster, and thus the throughput of an MRI system be improved, by splitting scans up and reordering the scan parts of a complete exam? The first results indicate that exams can indeed be executed faster by doing this. The research now focuses on the impact of the change of this architectural decision, and on determining by an extensive cost-benefit analysis whether this evolutionary step should indeed be implemented.

4.3 Connecting system and software

This research area focuses on the connection between system and software. The software integrates the different parts of the system, and it controls the mechanical, optical, electrical, and other components. Within this research area, we concentrate on two topics, where the software helps us to increase our understanding of the system as a whole.

The first research topic concentrates on the hidden dependencies in the software. Hidden dependencies are dependencies of which stakeholders, such as architects and software developers, are not aware. Hidden dependencies can be caused by people leaving the project, imperfect abstractions, and missing feedback loops. These hidden dependencies introduce risks and uncertainty in evolving the software, amongst others, since they are excluded from change impact analysis. Since dependencies can be both static and dynamic, source code analysis will not uncover all dependencies. In our research, we focus on dynamic analysis for uncovering hidden dependencies. By uncovering hidden dependencies, we make evolving the software better predictable and reduce the risks involved. Our first results indicate that the amount of hidden dependencies is considerable, and hidden dependencies are mostly dynamic dependencies.

The second research topic concentrates on obtaining insight in how software of systems evolves and the exploitation of this insight. In particular, we would like to determine what kind of information about the evolution of the software of a system adds value for its stakeholders. Furthermore, we would like to make this kind of information easily and intuitively accessible, e.g., by exploiting the hierarchy of the software and the power of hypertext (such as HTML). Our approach consists of many short iterations that involve the different stakeholders of the system. We started with the version control archive as our source of information about the evolution of software. We selected the version control archive, because already at CMM level 2 [9] the presence of such an archive is mandatory, and consequently most, if not all, software-intensive systems will have one. In a version control archive not only all versions of all files are recorded but also the meta-data about every change. This meta-data includes the person who made the change; the

files that were changed; and the time the change was put into the version control archive. Under the assumption that each change has one cause, such as the addition of a new feature, a bug fix, or a refactoring, files changed together are thus related. From the version control archive, one can collect statistics on the likelihood that two particular files changed together: an indicator of the strength of the relationship between those files. This information could have value for project managers and software developers, for example, in effort estimation and during the implementation of changes. However, presenting these relationships is not straightforward, since for software-intensive systems, the number of files typically exceeds 10,000. After a few iterations, we are convinced that these relationship strengths are indeed valuable for the system's stakeholders. The number of stakeholders involved in evaluating and fine-tuning our visualization tool is still increasing with every iteration step.

5. Summary

In this paper, we presented the applied research project Darwin. Darwin is a collaborative project between the Embedded Systems Institute, Philips Medical Systems, Philips Research, and five Dutch universities (Delft, Eindhoven, Groningen, Twente, and the Free University of Amsterdam). The Darwin project is executed using the industry-as-laboratory paradigm. The Darwin project addresses the real-world challenge: improve the evolvability of software-intensive systems.

Within the Darwin project a Magnetic Resonance Imaging (MRI) system is taken as an example of a software-intensive system. The focus of the project is on architecture and the research areas addressed are system engineering; modeling and analysis; and connecting system and software. The goal of Darwin is to understand evolvability as a system property; to identify, create, and apply constructs, models, and methods to support evolvability; to support the trade-off decisions the architect will have to make with respect to evolvability; and to support the sub-system and technology lifecycle view of a system. Some preliminary results have already been achieved and described, but a lot of work remains.

For more information about the Darwin project, please visit our project's website at www.esi.nl/darwin. When available, project results will be published on this website.

6. Acknowledgements

We would like to thank Wim Stut and Wim van der Linden for their useful remarks on an earlier version of this paper.

7. References

[1] P. America, D. K. Hammer, M. T. Ionita, H. Obbink, and E. Rommes, "Scenario-Based Decision Making for Architectural Variability in Product Families". In: R. L. Nord, ed.: Software Product Lines, Third International Conference, SPLC 2004, Boston, Massachusetts, August 30 - September 2, 2004, Springer, LNCS Vol. 3154, pp. 284-303.

[2] R. Arnold and S. Bohner, Software Change Impact Analysis, Wiley-IEEE Computer Society Press, 1996.

[3] C. Y. Baldwin and K. B. Clark, *Design Rules – The Power of Modularity*, MIT Press, March 2000.

[4] C. M. Christensen, R. Bohmer, and J. Kenagy, "Will Disruptive Innovations Cure Health Care?" *Harvard Business Review*, September-October 2000, pp. 102-112.

[5] T. Copeland and V. Antikarov: *Real Options, a Practitioner's Guide.* Texere, New York, 2003.

[6] E. Fricke and A. Schulz, "Design for Changeability (DfC): Principles to Enable Changes in Systems Throughout Their Entire Lifecycle", *Systems Engineering*, Vol. 8, No. 4, 2005, pp. 342-359.

[7] T. Herald, D. Verma, and T. Lechler, "A Model Proposal to Forecast System Baseline Evolution due to Obsolescence through System Operation", In Proceedings of 5th Annual Conference on Systems Engineering Research, Hoboken, New Jersey, March 14-16, 2007.

[8] N. Loughran, A. Rashid, "Supporting Evolution in Software using Frame Technology and Aspect-Orientation", Workshop on Software Variability Management, Groningen, The Netherlands, 2003, pp. 126-129.

[9] T. G. Olson, L. Parker Gates, J. L. Mullaney, J. W. Over, N. R. Reizer, M. I. Kellner, R. W. Phillips, and S. J. DiGennaro, "A Software Process Framework for the SEI Capability Maturity Model: Repeatable Level", CMU/SEI-93-SR-007, June 1993.

[10] D. L. Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules", *Communications of the ACM*, Vol. 15, No. 12, December 1972, pp. 1053–1058.

[11] D. Rowe, J. Leaney, and D. Lowe, “Defining systems evolvability-a taxonomy of change”, Proceedings of the International Conference and Workshop: Engineering of Computer-Based Systems (ECBS '98), Jerusalem, Israel, March 30th - April 3rd 1998, pp. 45-52.

[12] E. Suh, Flexible Product Platforms, PhD Thesis, Massachusetts Institute of Technology (MIT), USA, 2005.

[13] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, and H. Obbink, “Software Product Family Evaluation”, Software Product Lines, Lecture Notes in Computer Science, Vol. 3154, 2004, pp. 110-129.

[14] R. van Ommering, Building Product Populations with Software Components, PhD Thesis, University of Groningen, The Netherlands, 2004.