Coping with System Evolution -

Experiences in Reverse Architecting as a Means to Ease the Evolution of Complex Systems*

P. Daniel Borches **G. Maarten Bonnema**
Laboratory of Design, Production and Management,
Department of Engineering Technology, University of Twente
P.O.Box 217, 7500 AE Enschede
The Netherlands
p.d.borches@ctw.utwente.nl g.m.bonnema@utwente.nl

Abstract. Creating systems from scratch is time consuming and costly, therefore companies often choose to evolve existing systems. The understanding that a company has about the impact that a change has in the system architecture determines their ability to cope with system evolution.

System architects and designers need to have an architecture representation that enables them to understand and to foresee consequences of evolving the system. This representation however is often not documented. Reverse architecting enables to recover the architecture representation. In this paper, experiences in reverse architecting in a industrial case at Philips Healthcare MRI Group is presented. We show that the proposed approach provides an effective framework to reason about evolvability and impact that design changes has on the system.

Introduction

System requirements change over time; consequently, companies need to systematically evolve their products to cope with those changes. Since developing a system from scratch is time consuming and costly, new systems are often created by evolving an existing system (Suk suh et al. 2008). However, the effort and resources required to adapt complex systems to changing requirements can be significant; even minor top-level functional changes can have lengthy, costly and difficult to predict development cycles. If a change of requirements occurs, the effect can ripple through the entire system due to dependencies, known and hidden, in the system.

The area of system evolution comprises two main activities. First of all to anticipate (stepwise) for future developments of the system's context, and secondly to drive a system's architecture design in such a way that the ripple effect caused by changing requirements is kept small. In other words the effect of a change should be localized (Baldwin and Clark 2000). In this context system evolvability can be defined as: *a system's ability to adapt to changing requirements throughout its lifespan in a time-efficient and cost-efficient way* (Borches and Bonnema 2008). Hence, a company's ability to undertake and manage system evolution, can be influenced greatly by their

*This work has been carried out as a part of the DARWIN project at Philips Healthcare under the responsibility of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economics Affairs under the BSIK program.

understanding and knowledge of the impact that a change in the architecture has on the overall design and implementation. This statement is supported by authors such as (Isaac and McConaughy 1994; Steiner 1998).

This paper reports an approach to cope with system evolution and observations from its application in a real industrial case of a complex system that is being evolved; a Magnetic Resonance Imaging (MRI) system. By understanding the barriers and problems that arise when designs and their implementations evolve, and by seeing these consequences firsthand, general statements can be extracted from the specific use case.

In following section, previous work regarding evolution of complex systems and how architectures overviews created by reverse architecting can be used to support evolution is presented. In section 3 we discuss our experiences in reverse architecting a system by discussing our experiences from an industrial use case. In section 4 the evaluation of the work is presented. Finally, section 5 presents the conclusions and future work.

Industrial case: Magnetic Resonance Imaging (MRI) scanner

Magnetic resonance imaging (MRI) is an medical imaging modality that detects small changes in the magnetism of the atom's nucleus. An MRI system requires a multidisciplinary design team with competences in areas such as mechanics, electronics, physics, material science, software and clinical science. Typically people are specialized in a single discipline, and each discipline uses its own vocabulary. This adds to the complexity of the design process. Besides this, all the disciplines have to work together on different aspects of the design, such as real time behaviour, non-real time behaviour, control theory, analogue and digital technology, power related issues such as cooling and aspects such as user system interaction.

To illustrate the complexity of the MRI systems, some indicative numbers of Philips Healthcare MRI products and its development are provided in Table 1. All this is excluding research, marketing and so on. In addition to these numbers, the MRI process is in itself very difficult and involves many parameters and several domains (Weishaupt, Köchli, and Marincek 2006).

Parameter	Value
Developers	~400
Disciplines	Physics, Mechanics, Electronics, Software, Medical applications etc.
Development sites	3
Subsidiary sites	All around the world
Technologies	~50
Lines of SW code	$7*10^6$ (~10 different languages)

Table 1: Characteristic parameters of the MRI system development.

Since Philips released the first commercial scanner back in the 80's, it has been redesigned several times leading to the present system. The fundamental architecture of the system has remained almost unchanged compared to the original system. Therefore an MRI system is a very suitable use case to study the evolvability of large and complex systems.

Evolution of complex systems

Evolution as a concept has its deepest roots in the biological and social sciences. Darwin's theory, characterized by heritable variation and natural selection, is often used as a starting point (Darwin 1859). Biological and social approaches are however not directly applicable for complex systems, as species and systems are not similar enough (e.g. there is no clear analogue of 'gene' for artificial entities).

In technical sciences such as computer science, significant research has been carried out exploring the relation between design and evolution (MacCormack, Rusnak, and Baldwin 2008). This is due to the fact that software projects rarely start from scratch but rather prior versions are used as a platform. In the system field, many years have passed since the first papers regarding 'system evolvability' was published (Simon 1962; Isaac and McConaughy 1994); some theoretical work has been done in the field (Rowe and Leaney 1997, 1998; Christian III 2004); a few papers attempted to measure evolvability of complex systems (Christian III 2004; Christian III and Olds 2005), however they base those measures on expert's estimations rather than on the system itself; few papers tried to theoretically analyze evolvability on software systems (Rowe and Leaney 1998; Christian III and Olds 2005), yet the theoretical models proposed require to know both the initial and evolved system and they are too abstract to be applied in industry.

The importance of adopting evolvability has been discussed by several authors (Isaac and McConaughy 1994; Steiner 1998; Rowe and Leaney 1997; Christian III and Olds 2005; Ring and Fricke 1998) and its role in the system architecture has been described in (Isaac and McConaughy 1994; Steiner 1998). Outside the software field however, evolvability is almost an unexplored field; evolvability definition is still open for discussion (Christian III 2004; Christian III and Olds 2005; Rowe and Leaney 1997, 1998); there is no formal way to assess or measure evolvability (Christian III 2004; Christian III and Olds 2005); evolvability is confused with other system's properties (Fricke and Schulz 2005) and there is almost no real work done in the field. That is to say, although attempts were made in the past, it can be concluded that no satisfactory formal way to assess system evolvability exists and designing for evolvability is usually delegated to the designer's intuition.

In recent years, the design of systems has been receiving more attention both in industry and academia. An important stream of research focuses on the relation between system design decisions and system evolution. It is argued that some designs are more "adaptable" than others, in that they facilitate the process of modifying or updating the system's components to reflect changing conditions (MacCormack, Rusnak, and Baldwin 2008). The degree to which system's components may be separated and recombined and the degree to which the system architecture enables or prohibits the mixing and matching of components, is usually referred to as modularity (Schilling 2000). Modularity helps to characterize different product architectures and to incorporate guidelines of evolution or other system properties to a design. Modular designs are loosely-coupled in that changes made to one module have little impact on the others. The link between modularity and evolution was first made explicitly by Simon (Simon 1962), who argued that "nearly-decomposable" systems facilitate experimentation and problem solving. Recent work formalizes this reasoning by showing that modularity creates design "options" (Baldwin and Clark 2000). Within a system, modules are free to evolve in an independent manner; hence greater modularity is associated with an increase in the potential of system evolution.

In order to identify and create those modules, the key task is to understand how changes in the architecture would affect the overall system. By understanding and assessing the consequences of changes, design effort can be directed towards avoiding undesired impacts and guiding the design of the system to the one that enables easy evolution Then, an evolvable system could be built on the aspects of the system which are most likely to remain unchanged (Percivall 1994). By studying change over history predictions about change can be made. These can be formalized that to make the rest of the system to evolve easier, such as using popular standard components to the “often changing” parts.

Managing complexity

As complexity is widely studied (see for instance (McDermid 2000; Gell-Mann 1995; Axelsson 2002; Ottino 2003), it is not our object to repeat these. One approach to reduce complexity when representing a complex system is by encapsulation (Mattos, Meyer-Wegener, and Mitschang 1993). Encapsulation can be used to group certain pieces of functionality so that they can be treated as a "black box". Often such a black box carries the name of its main function ("power supply", "position sensor", etc.). This reduces the number of individual components to take into account and therefore helps to keep the big picture. This approach is useful as long as the limitations, in particular side-effects, are known. While the design project moves forward, each black box will be filled in with new, smaller, black boxes or real-life components. This inevitably produces side-effects and parasitic effects, and modified demands on the infrastructure. The problems that are associated with these effects are in general not communicated to the next higher hierarchical level. Using a “grey box" that does expose some of its internals would be a better approach (Bonnema and Borches 2008). This is essential in order to highlight those critical parameters that should be taken into account when evolving a system, that might otherwise not be present at a high abstraction level.

Reverse architecting approach

One aspect of Systems Engineering involves transforming an operational need or market opportunity into a system description to support detail design. For an existing system though, this system description and detail design (should) already exist. It is however often not documented nor explicit, which makes it difficult to adapt the system to new needs. An approach to recover and represent the system description is needed in order to effectively evolve a system.

The approach we propose to cope with system evolution is shown in Figure 1. This approach to identify and create modules in a system to enhance the system evolvability, bases on literature and our observations from industry. To identify and create modules, insight and understanding of the system is required. Unfortunately these insights and understanding are often not documented. This means that the system architecture representation has to be, largely at least, reconstructed. Doing this is called reverse architecting. As stated in (Krikhaar 1997); *reverse architecting is a flavor of reverse engineering that concerns all activities for making existing architectures explicit, and the main goal of reverse engineering is to increase comprehensibility of the system for maintenance and new development*. The MRI as a complex system is the creation of a multidisciplinary team. Yet we believe the lessons from reverse architecting in software engineering (Mayrhauser, Wang, and Li 1999) and in building architecture (Galal-Edeen 2002) apply to the systems engineering discipline.

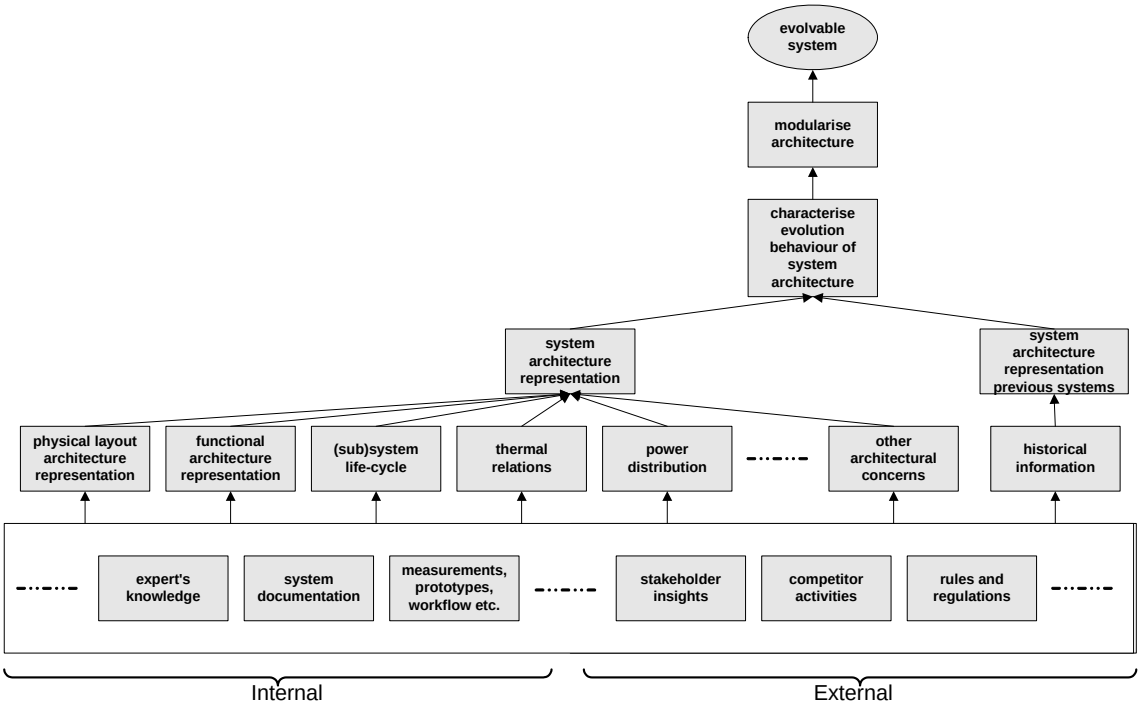

Figure 1 Approach to design an evolvable system (feedback loops not shown).

In (Muller 1996), it is mentioned that the reverse engineering of a system consists of three phases; **information extraction**, **abstraction** and **presentation**. Although not explicitly mentioned, it is clear that frequent feedback loops among those phases are required in order to incorporate new insights and findings discovered during the process. To create the system architecture representation, it is important to reconstruct relevant architectural concerns related to evolvability. That is to say, to make system aspects critical for the system evolution explicit and to represent them in such a way that they are taken into account during the evolution of the system.

For the abstraction and presentation phase, the concept of an **architecture overview** is introduced. The main goal of an architecture overview is to have a manageable architectural representation of the system that enables system architects and designers to understand the consequences of system evolutions at the system level (Borches and Bonnema 2008). An architecture overview shall help to provide a broad, comprehensive and easy to handle view of the system, and to highlight architectural concerns of the system under study.

Unfortunately, formal theories are not suitable for formulating design rules when designing good architecture overviews. It is difficult to determine what makes a good architecture overview. It is not without reason that there are so many definitions and statements on architectures. A useful architecture overview has different views of the system (Muller 2006; Zachman 1987). The most common view is probably a block diagram consisting of physical components and their interfaces. For complex systems however, other views are equally or even more important. As supported by (Shaw 1989), other high-level abstraction views that are independent of the components - such as a functional view - should also be included.

Experts from all fields are familiar with physical views and similar models, however when thinking in functions they seem to be confused. This is partly due to the fact that the term “function” can be interpreted in a variety of ways. Nevertheless, the notion of function as defined in (Blanchard and Fabrycky 1998) (“A specific or discrete action

that is necessary to achieve a given objective") has proven to be very useful in the design process (Bonnema and van Houten 2006). In addition to the abstraction and general reasoning that is possible by using functions, in complex systems design, function-modelling has the advantage that functions do not change so much during the life of the system. Therefore function models can be used as link between different physical views of different evolutions of a certain system.

Finally, as shown in Figure 1, once the architecture representation is recreated, we can compare the current system architecture representation to that of historical versions of the same system. Then, by taking into account expectations of the future developments in the system's context, experts can use them to identify parts of the system that could be modularized to reduce the impact of change upon a requirement change, thereby increasing the system's evolvability.

Industrial Case studies

The work presented here has been conducted as part of the Darwin project (Laar et al. 2007) within the Product Group MRI of Philips Healthcare, by using the paradigm "industry as a laboratory". We have obtained first-hand information on the system architecture by participating in different development projects.

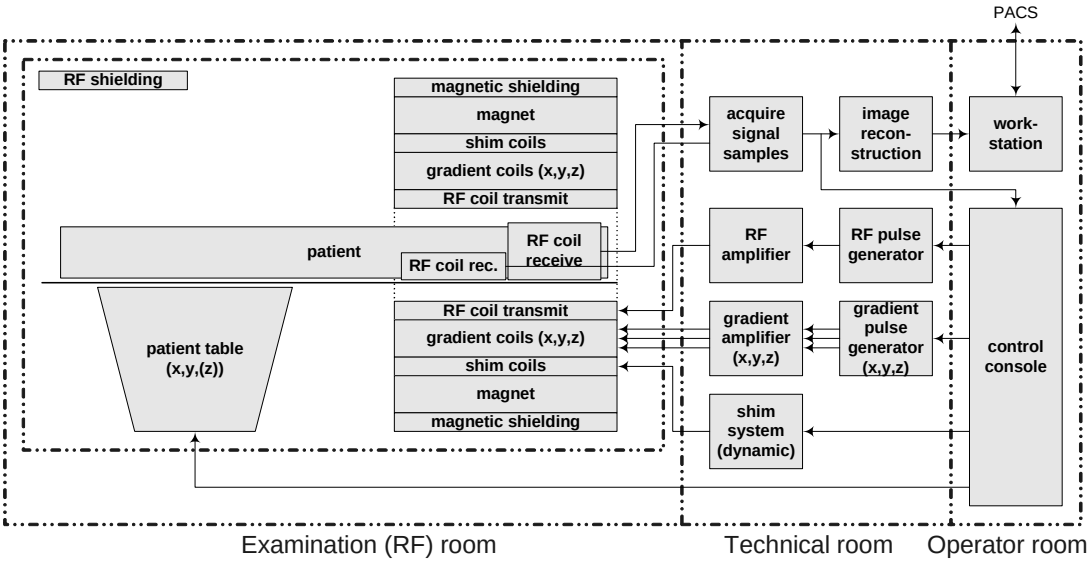

Figure 2 MRI system overview

Figure 2 shows an overview of the MRI system. The cylindrical layers of the machine, and the opening, called bore, with the patient and receive coils inside are drawn as a cross section. The biggest component of the MRI system is the magnet, which provides the static magnetic field. This static magnetic field suffers from inhomogeneity. To minimize this, shimming is applied. Passive shimming involves placing pieces of steel into the magnet, and is done at install time of the MRI system in a hospital. Dynamic shimming, by controlling currents in electrical shim coils, can be performed on a per-patient or exam case. The magnetic field generated by the magnet outside the bore area is called fringe field, and is limited by shielding. Three orthogonal linear magnetic field gradients (x, y, z) are used for spatial localization of the MRI signal. To generate pulses of current at a specific frequency (Larmor frequency), the RF transmit coils are used. Depending on the examination to be performed, the control console determines the signals needed to create RF pulses and gradient pulses. To have the desired effect, these pulses need to be amplified considerably, and this is done in the RF amplifier and

gradient amplifier respectively. The positioning of the patient in the bore is also done by the control console and based on the exam to be performed. The RF receive coils detect the MRI signal, which is digitized for use in either the image reconstruction or for control purposes, such as power calibrations. A physician or radiologist can view the reconstructed image on the workstation, and when the result is satisfactory it can be sent to the picture archiving and communication system (PACS) for future use[2].

Study case: Communication architecture evolution

The first author investigated the evolution in the communication architecture of the MRI system. A new communication design was desired to cope with increased performance and to cope with technology obsolescence due to proprietary development of communication elements. To design a new communication architecture poses great difficulties as communication requirements in a MRI system are difficult to meet due to the high performance required. As an example; synchronization among subsystems (which in some cases are separated several meters) is in the order of nanoseconds. Reliability of communications is extreme due to safety reasons. Low latency is required to collect patient data timely. Real time monitoring is needed, etc. Additional constraints need to be taken into account, as the system is located in presence of a powerful electromagnetic field which may have an effect on the components used (MRI compatibility). Understanding and allocating all those communication requirements to specific components and functions is one of the major concerns. As requirements came from different disciplines involved, the rationale of those requirements is not always clear to all team members.

Goal

In order to support the evolution of the communication architecture, the goal was to provide the experts with means to easily discuss alternative communication designs and requirements, enabling them to foresee potential impacts of those alternatives.

Reverse architecting process

Even in large companies, complex systems are typically poorly documented. The main architecture knowledge resides in the expert's minds, and only part of that knowledge is documented. Some key knowledge regarding the system architecture and design decisions may be lost, especially in long-lived systems, due to experts leaving the company, while design decisions have not been documented and so on.

As shown in Figure 3, the process we used to obtain first-hand information regarding the system's architecture was mainly through interviews with key personnel in the organization. In addition, company documents covering topics such as system requirements and specification, building block architecture, and system architecture were collected and reviewed. Extracting the essentials from those sources was key to create the architecture overview. It was known that complete consistence of the architecture knowledge was an illusion; there is always some degree of uncertainty that will be reflected in the architecture overview (this uncertainty was reduced as new insights and knowledge were available).

[2] For more detailed information on MRI systems and technology see (McRobbie et al. 2007; Weishaupt, Köchli, and Marincek 2006)

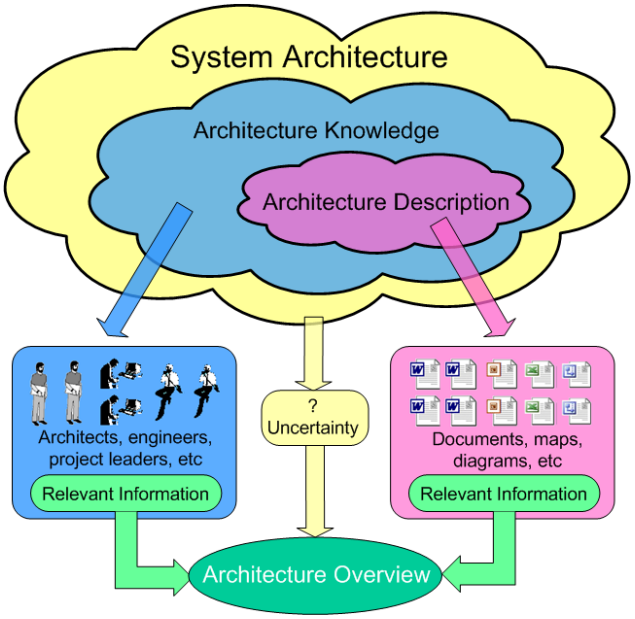

Figure 3 Sources of architecture information

Little was known beforehand by the authors regarding the MRI system. During the information extraction phase, documents were reviewed to get the basic domain knowledge of the MRI. As shown in Figure 4 a), a view showing how the customer may perceive the real system was created to be used as a starting point. From this view, a room layout was selected for the physical view. The few functions identified during the document research were used to create the initial functional view. These views combined resulted in an architecture overview draft that was used during the rest of the process. Thereafter, in order to retrieve architectural knowledge from the experts, interviews with key personal were conducted. The following approach was used:

1. Expert selection: A meeting with the main system architect and the project leader was held to discuss which key personnel should be interviewed to complete the architecture overview.
2. Goal description: Prior to the interview, an introductory text was sent to the interviewees explaining the reason and topic of the interview. The concept of “architecture overview” was briefly explained in general terms to provide a context for discussion. This was necessary to avoid misunderstandings regarding the many connotations of the “architecture overview” concept, which may otherwise have resulted in an unfocused interview. The introductory text also allowed the interviewees to prepare themselves.
3. Preparation: To take advantage of the expert’s knowledge, it was important to previously become familiar with the topic under discussion. Relevant documentation was reviewed and analyzed, and key questions were prepared.
4. Questioning: During the interview key questions were asked. The architecture overview is used to guide the expert to get the desired outcome; it was much easier for the expert to modify a wrong architecture overview than creating a new one from scratch. Finally, the interviewer finished asking for additional output such as relevant documentation or further contacts.
5. Update: After the interview the information provided was reviewed and changes were incorporated to the architecture overview. Feedback was requested from the expert in order to validate the new information and clear up inconsistencies.

6. Completeness check: The steps 2-5 were repeated until the architecture overview was stable enough.
7. Validation and acceptance workshop: As the overview had to be used by a different set of stakeholders such as designers, architects, engineers and so on, all of them should feel that they have contributed to its creation. For this, a workshop was conducted to include their main concerns and insights in the final architecture representation.

In the case presented, the creation of the architecture overview took approximately 4 man-months. About 20 experts were interviewed (some of them more than once). Mostly, only individuals were interviewed to increase the possibility for having in-depth discussions and ensuring that the interviewee could speak freely.

Outcome

Mainly three views were produced for this study case (detailed views are abstracted due to confidentiality reasons); a physical view, a functional view and a view containing requirements and quantification of key design parameters (not shown).

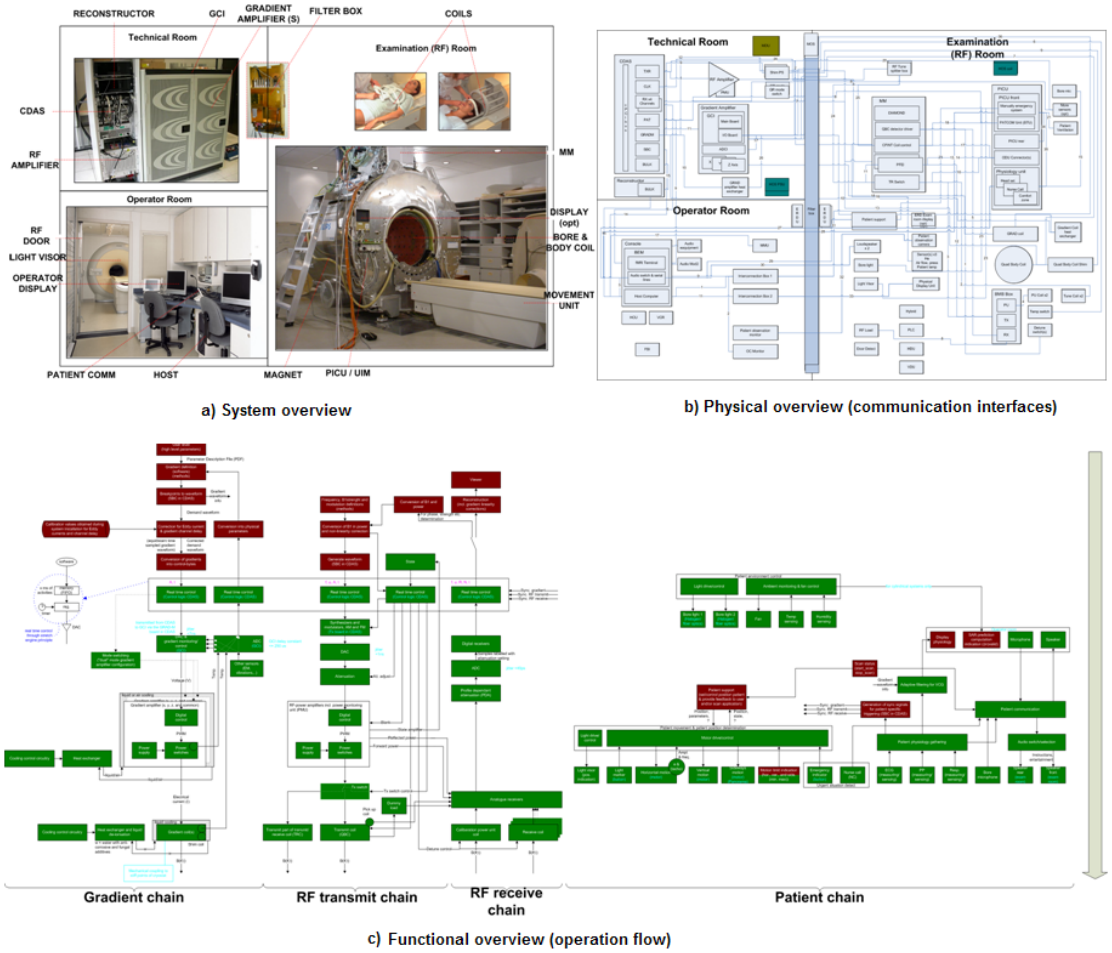

Figure 4 MRI architecture overview; a) System overview; b) Physical overview (communication interfaces); c) Functional overview (operation flow)

Figure 4 b) shows the physical overview with the main MRI building blocks arranged by spatial location, and the communication interfaces. This representation is a logical choice for the MRI, as the communications requirements depend on which room the components are located (e.g. the requirements in the examination room are different

than for the operator room due to the magnetic field). The functional view in Figure 4 c) shows the functions performed by the MRI system to create an image, from the moment the user enters the necessary information (top of the diagram) till the image is created (bottom of the diagram). The functions are arranged horizontally by chains (subsystem division of the MRI system, according to their functionality and organization structure), and vertically according to the moment the function is performed. The functions are colored based on whether they are allocated to software (red) or hardware (green) components. Arrows represent inputs and outputs. As we are using the "grey box" concept described in the previous section, additional relevant information (functional and non-functional) is included in the diagrams (e.g. maximum communication delay allowed for a specific function or component).

The MRI architecture overview shown was used during the whole process as a baseline for discussion. The views produced are not isolated from each other, functions were allocated to physical elements and requirements were also allocated to the views. When necessary other architecture overviews were derived from this view highlighting specific aspects of the system such as real time control of communication(see Figure 5).

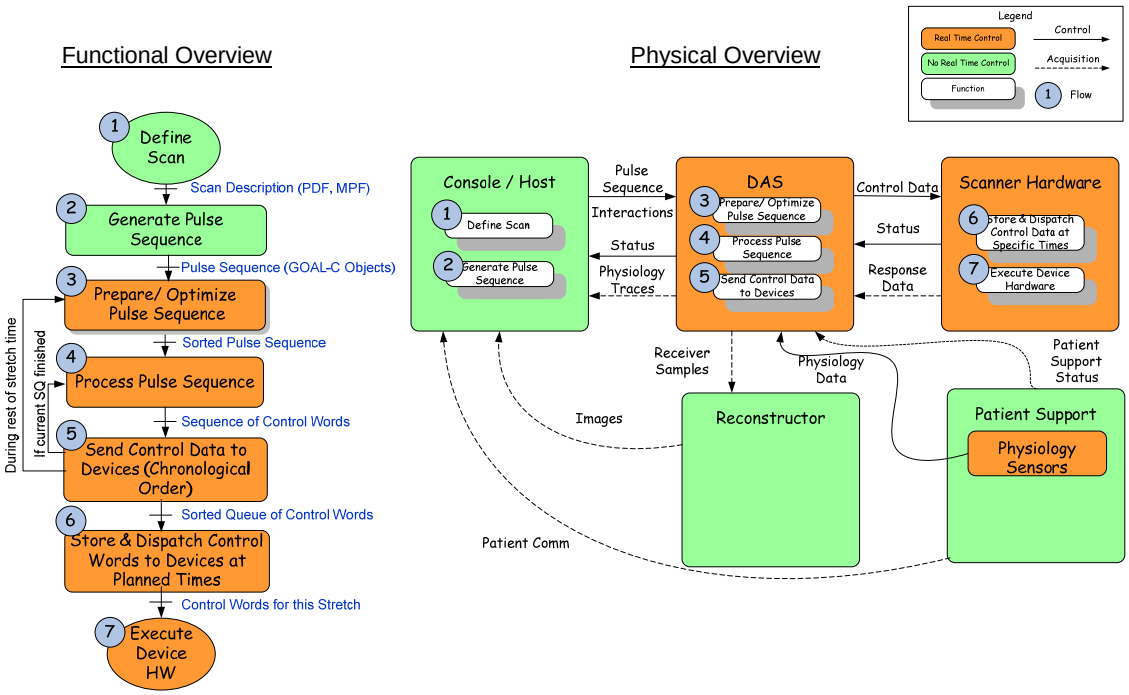

Figure 5 Top level architecture overview for real time control of communication (function allocation explicit)

When evaluating alternative communication designs and technologies, the architecture overview was used to identify potential changes to functions or physical elements. It was also used to understand the rationale of critical communication requirements by using the overview as guidance to relate elements and functions that conforms the requirement. E.g. starting from the bottom of the functional view, the bandwidth required by the first function of the RF subsystem (collect RF signal from patient) was extracted. Going up in the functional view this would mean that the upper function (sampling RF signal) would increase the required bandwidth, and so on. Following the same approach, as functions are allocated to physical elements, constraints such as technology limitations of those physical elements were also taken into account.

Consequences of architectural changes

It is well-known that the task of estimating change impacts in complex systems is one of the larger engineering problems (Ring and Fricke 1998; Clarkson, Caroline, and Claudia 2004). We do not aim to provide a precise method to estimate detailed component change, but a structured approach to foresee potential impacts to the architecture by using architecture overviews.

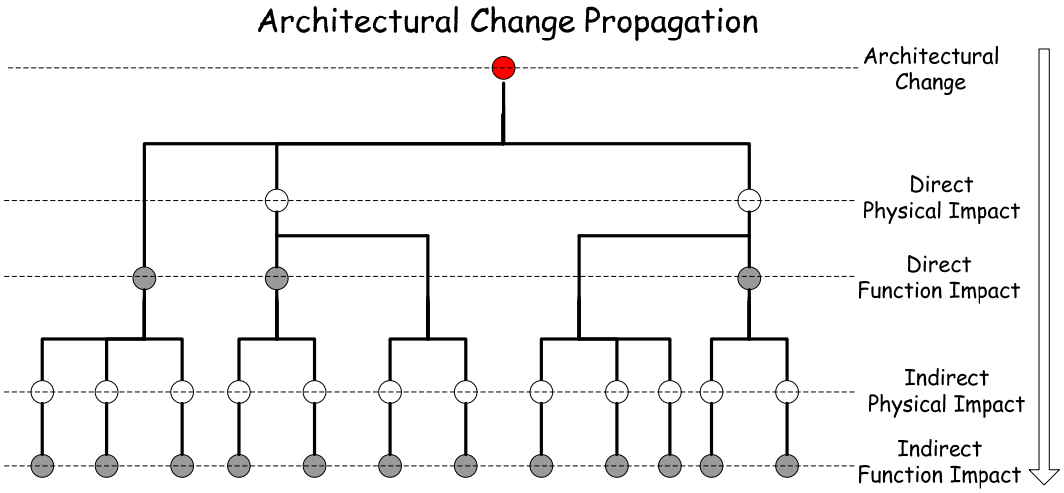

Figure 6 Estimation of Architectural Change Propagation

As shown in Figure 6, by using the architecture overview the impact of changes can be estimated by following the potential propagation changes of linked elements. E.g. an architectural change such as a redesign of the "collect RF signal" function could mean that a modification (direct impact) in the coil element (where the function is allocated) may be needed, as well as an impact in the functions that depend on it. Indirect impact to other parts of the architecture can be traced by repeating this approach. Although evolution history can provide insight regarding whether or not the propagation of a specific path will take place, we have found that experienced architects are usually the most reliable source to get this information.

It could be argued that evolvability measures could be extracted from this approach (e.g. by assigning likelihood and impact numbers to potential propagation paths and then calculating risk estimations of architectural changes), however we have found that assigning numbers to all potential propagation paths requires considerable effort and the numbers obtained are usually not trusted by experts due to the many assumptions made.

Study case: New coil functionality

To validate the approach it was also applied to a different study case at the coil department of the MRI division. A new design for the MRI coils was required as part of a new project therefore a new set of requirements was necessary. Along with the requirement engineering process, a coil architecture overview was created. Experts and stakeholders were asked to contribute both to the requirement gathering process and to complete the architecture overview. The same approach described in the previous study case was used to create the architecture overview, thus we do not repeat it.

Evaluation of the work

The main value of the architecture overview is that it improves communication and discussion among disciplines when evaluating proposed changes. The architectural representations of different aspects of the system have proven to eliminate the jargon among disciplines, facilitating discussion. It was especially useful when discussing requirements about the next system evolution, and to easily identify which physical elements or functions shall be inherited or replaced in the next system. The stability of the functional overview was proven during this work. Less than 10% of the functions of the system have been added or replaced within three generations of the MRI system (a time window of more than 10 years). On the other hand, 60% or more of the physical elements have been replaced or redesigned in the same period, leaving old physical views outdated.

Feedback from experts using the architecture overview was more rapidly obtained and more specific than compared with other forms of communication. As an example; feedback on alternative communication designs was asked to 20 experts in the form of questions related to a requirement specification (RS) document created to describe the changes required in the system (approximately 60 pages). This is the traditional way of discussing system changes within Philips Healthcare. Less than 20% of the experts found time to read the document and provide feedback. In contrast, meetings that were held using the architecture overview to ask for feedback regarding the same issues, resulted in all 20 experts providing feedback.

The architecture overview serves as a repository of architectural knowledge. It describes the architecture baseline, from which systems are derived. It acts as the system collective memory. If the system model is unknown (undocumented) it will be difficult to cope with future evolutions with reasonable resources. This is especially true in systems such as the MRI with little production volume, as there are almost no publications on architectures for diagnostic systems. The general publications available are too abstract and can often be interpreted in a variety of ways, thus being not specific enough for our application.

The proposed approach provides a framework to reason about evolvability and impact of design changes. Although acceptance by developers of the architecture overview was sometimes a challenge, due to the lack of low level detail in the views, system architects and project managers on the other hand, realized the utility of having the architecture overview, and hung those views on their walls for daily use. It was stated that in order to have a true impact in the design process, the architecture overview should become part of the system description, as some of the information contained in the architecture overview was not explicit anywhere else.

Finally, during a formal presentation of the work at the company, it was found that new employees could benefited greatly from the architecture overview. It is estimated within Philips Healthcare that around 5 years of working experience within the MRI division is needed to become familiar with all aspects of the system. This results in new employees working on different issues without really knowing the impact that their work might have on the overall system. Several new employees stated that having those views would reduce their learning curve.

Conclusions and Recommendations

Companies need to systematically evolve their products to cope with those changes. Since developing a system from scratch is time consuming and costly, new systems are often created by evolving an existing system. However, the effort and resources required to adapt complex systems to changing requirements can be significant.

Greater modularity is associated with an increase in the potential of system evolution. In order to identify and create those modules, the key task is to understand how changes in the design would affect the overall system. By understanding and assessing the consequences of changes, design effort can be directed towards avoiding undesired impacts and guide the design of the system to the one that enables easy evolution.

To identify and create modules, insight and understanding of the system is required. Unfortunately these insights and understanding are often not documented. This means that the system architecture representation has to be, largely at least, reconstructed. Doing this is called reverse architecting.

The evolution process in a complex system requires the participation of a multidisciplinary team. Different backgrounds and specific jargon obscure the communication process, leading to misunderstandings and lack of clarity. To reduce these barriers, the concept of an architecture overview is introduced. The main goal of an architecture overview is to have a manageable architectural representation that enables system architects and designers to understand the consequences of system evolutions at the system level.

A view that must be included in the architecture overview is a functional view, which is known to be useful in the design process. The main barrier to use it is that many experts are not used to think in functions, and there is not a clear definition across disciplines of the term function. More emphasis on functional thinking and modeling should be necessary during the education of engineers.

Acknowledgments

The research is conducted in cooperation with the MRI group of Philips Healthcare. We thank all the interviewees from Philips Healthcare for their support and contributions. We want to thank specially Phil van Liere and Alexander Douglas for the time dedicated to this project. We also want to thank the support of Pierre van de Laar, Gerrit Muller and other members of the Darwin project.

References

Axelsson, J. 2002. Towards an Improved Understanding of Humans as the Components that Implement System Engineering. *12th International Symposium of the International Council of Systems Engineering, Las Vegas*.

Baldwin, C. Y., and K. Clark. 2000. *Design rules: The power of modularity*. Cambridge: MIT Press.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 1998. *Systems Engineering and Analysis*. Third ed. Upper Saddle River, New Jersey: Prentice Hall.

Bonnema, G. Maarten, and Pedro Daniel Borches. 2008. Design with Overview – how to survive in complex organisations. In *18th Annual International Symposium of INCOSE (IS2008)*. Utrecht, The Netherlands: INCOSE.

Bonnema, G. Maarten, and Fred J.A.M. van Houten. 2006. Use of Models in Conceptual Design. *Journal of Engineering Design* 17 (6):549--562.

Borches, P. D., and G. M. Bonnema. 2008. 'Living' Architecture Overviews - Supporting the Design of Evolutionary Complex Systems. Paper read at CIRP Design Seminar, at Twente, The Netherlands.

———. 2008. On the Origin of Evolvable Systems: Evolvability or Extinction. *Proceedings of the TMCE* 2:1351-1353.

Christian III, John A. 2004. A Quantitative Approach to Assessing System Evolvability.

Christian III, John A., and John R. Olds. 2005. A Quantitative Methodology for Identifying Evolvable Space Systems. In *1st AIAA Space Exploration Conference January 2005, Orlando, FL*. Orlando, FL.: Georgia Institute of Technology.

Clarkson, P. John, Simons Caroline, and Eckert Claudia. 2004. Predicting Change Propagation in Complex Design. *Journal of Mechanical Design* 126 (5):788-797.

Darwin, C. 1859. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. Edited by J. Murray. London.

Fricke, Ernst, and Armin P. Schulz. 2005. *Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle*. Vol. 8: John Wiley and Sons Ltd.

Galal-Edeen, G. H. 2002. Reverse architecting: seeking the architectonic. Paper read at Ninth Working Conference on Reverse Engineering

Gell-Mann, Murray. 1995. "What is Complexity?" *Complexity* 1 (1):16--19.

Isaac, D. , and G. McConaughy. 1994. The Role of Architecture and Evolvability Development in Accommodating Change. *INCOSE'94*:541-545.

Krikhaar, R. L. 1997. Reverse architecting approach for complex systems. Paper read at IEEE International Conference on Software Maintenance, at Bari.

Laar, Pierre van de, Sjir van Loo, Gerrit Muller, Teade Punter, David Watts, Pierre America, and Joland Rutgers. 2007. The Darwin Project: Evolvability of Software-Intensive Systems. *23th IEEE International Conference on Software Maintenance (ICSM 2007)*.

MacCormack, A., J. Rusnak, and C. Baldwin. 2008. The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry: Harvard University.

Mattos, Nelson M., Klaus Meyer-Wegener, and Bernhard Mitschang. 1993. Grand tour of concepts for object-orientation from a database point of view. *Data & Knowledge Engineering* 9 (3):321--352.

Mayrhauser, A. von, J. Wang, and Q. Li. 1999. Experience with a reverse architecture approach to increase understanding. Paper read at IEEE International Conference on Software Maintenance, at Oxford.

McDermid, J. A. 2000. Complexity: concept, causes and control. Paper read at Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000).

McRobbie, D. W., E. A. Moore, M. J. Graves, and M. R. Prince. 2007. *MRI from picture to proton*. Edited by C. u. Press.

Muller, G. *How to Create an Architecture Overview* 2006 [cited. Available from www.gaudisite.nl.

Muller, H. A. 1996. Understanding software systems using reverse engineering technologies research and practice. Paper read at Tutorial 18th ICSE at Berlin.

Ottino, J.M. 2003. Complex Systems.

Percivall, G.S. 1994. System Architecture for Evolutionary System Development. In *Proceedings of the 4th Annual Symposium of the National Council on Systems Engineering*. San Jose.

Ring, J., and E. Fricke. 1998. Rapid Evolution of All Your Systems - Problem or Opportunity? *Proceedings of IEEE 17th DASC, Seattle*.

Rowe, David, and John Leaney. 1997. Evaluating evolvability of computer based systems architectures - an ontological approach. *IEEE Proc ECBS*:360-367.

———. 1998. Defining systems evolvability - a taxonomy for change. *Proceedings of the International Conference and Workshop: Engineering of Computer-Based Systems (ECBS '98), Jerusalen, Israel*:45-52.

Schilling, M.A. 2000. Towards a general modular systems theory and its application to inter-firm product modularity. *Academy of Management Review* 25:312-334.

Shaw, Mary. 1989. Larger scale systems require higher-level abstractions. *Proceedings of the 5th International Workshop on Software Specification and Design.*

Simon, H. A. 1962. The Architecture of Complexity. Paper read at American Philosophical Society.

Steiner, R. 1998. Systems Architecture and Evolvability - Definitions and Perspective. *Proceedings of the 8th Annual Symposium of the International Council on System Engineering*.

Suk suh, Eun, Michael R. Furst, Kenneth J. Mihalyov, and Oliver L. de Weck. 2008. Technology Infusion: An Assessment Framework and Case Study. Paper read at International Design Engineering, at New York, USA.

Weishaupt, D., V. D. Köchli, and B. Marincek. 2006. *How does MRI work*. Edited by Springer-Verlag.

Zachman, J. 1987. A framework for information systems architecture. *IBM Systems Journal* 26:3.

Biography

P. Daniel Borches received his Master's degree in Telecommunication Engineering from the University of Madrid Carlos III in 2004. He has worked several years in companies such as Telefónica R&D and Nokia. From 2006 he is working at Philips Healthcare Netherlands while doing his Ph.D. at the University of Twente. The main focus of his research is Systems Engineering and Systems Architecting applied in the industrial sector.

G. Maarten Bonnema is an assistant professor at the Laboratory of Design, Production and Management of the Faculty of Engineering Technology at the University of Twente. He studied Electronic Engineering, and did a two year designer course on Technical Systems. He has worked as a Systems Engineer at ASML. In 2006 and 2007 he was involved in the design of a wafer stepper at MAPPER lithography (part time). His research involves supporting system designers, conceptual design and complex design.