



Software Evolution and Roadmapping: A “Technology Ecosystem” Based Approach to Study Software Evolution Patterns and Drivers for Software in Medical Equipment

**Jacco Wesselius
Jan Willem van den Beukel
Wim Pasman**

Philips Medical Systems, March 2007

Abstract

Software Evolution is the process of repeatedly improving software in small steps. In many cases, software evolution is associated to agile development. The short term focus in most of the evolutionary and agile software development processes seems to conflict with the need for long term planning and roadmapping in industries with a long product life cycle. This is also the case when software development has to be well aligned with hardware and system design projects that use a more traditional, linear development approach.

In this paper, evolution patterns of medical equipment in Philips Medical Systems have been studied. Using the technology ecosystem approach proposed in [3, 4], this paper discusses the evolution patterns and the innovation drivers of these systems. The outcome clarifies the issues to be addressed, when using evolutionary development in this environment: planning of software evolution needs to be integrated in the technology and business roadmapping processes.

1. Introduction

Software Evolution denotes the process of developing software initially and repeatedly extending and improving it. As such, software evolution can be seen as the maintenance lifecycle phase. Software Evolution can refer to much more than the maintenance activities. It can also refer to the way software is developed in a series of small incremental/evolutionary steps. In that model, each step in the evolutionary software development life cycle changes the properties of the software product; either by adding functionality or by improving one or more of the quality attributes of the software product.

The evolutionary software development approach is tightly coupled to:

1. A quantitative approach to software requirements definition and software project planning and tracking [1] based on active measurement of the quality attributes of the software product in each evolutionary development step. The key element of the evolutionary software development approach is that it enables frequent measurements, which are vital to achieving a stable control loop.
2. An agile development approach [2] focussing on small incremental development steps providing value to the stakeholders in each individual step. In this approach, the focus is on exploration of stakeholder value, and on



optimisation of the communication amongst team members and the stakeholders.

The two properties of evolutionary development combine well into a development approach based on quantified project tracking (based on software quality properties rather than artificial project milestones) and optimal communication about product value with the stakeholders of the software development project.

In the ITEA Serious project, a consortium of commercial companies and academic institutions are studying the value of evolutionary software development, and are seeking for ways to optimise the application of the evolutionary development approach in commercial software development. The issue we address in this paper is one of the issues identified in the ITEA Serious project:

- Often large software products have a relatively long life cycle
- This is especially the case when the software is part of a large, complex and expensive hardware device. But it is also applicable to software that is closely integrated with the processes in a large and complex administrative organisation.
- This means that longer-term planning processes (roadmapping) are asked for to manage the value of the software in both installed base maintenance/service and initial developments or extensions of existing products.
- How to combine the rigid longer-term planning and control processes with agile and evolutionary software development processes?

Since we have been studying this question from our background in medical systems, we add one specific aspect to it: if software is part of a complex medical device (like an MRI-scanner, or an X-Ray Cathlab), its development is tightly coupled to and restricted by the development of the overall system. The development of the system and hardware context of the software is often carried out in a rigid, linear development process. Cost of materials and long lead-time items give a special dimension to system and hardware development, which makes it less suitable for the agile or evolutionary approach.

The question we have studied specifically is:

- To which extent can the embedded software development be approached as an evolutionary in the context of system and hardware development, which are mostly done in a linear manner?
- To which extent is the evolution of software independent from the development of its system and hardware context? What are the main innovation drivers that influence the evolution of the embedded software?

To study these questions we have used a model for technology evolution defined by Adomavicius et al from Carlson School of Management, University of Minnesota [3, 4]. This model provides a way for mapping the relations between the innovations in a product, its components and its environment. We have applied the model to four types of products in Philips Medical Systems:

- The software in an MRI-scanner
- The software in an X-Ray Cathlab
- The software in components that are part of X-Ray Devices: Image Processors



- The software component platform used in several Philips Medical Devices. These software components range from system specific software components to generic software components.

In this paper, we present the results of our study. First we shortly introduce the model we have taken from literature. After that we will discuss the way the innovation of our products maps on this model. This section will show what drives the technical innovation in the four products we used for our study.

After making this mapping, we realized that we needed an additional step in order to explicitly put the software component of our products on the map. We made a small extension to the model from Adomavicius et al, and mapped the evolution of the software in our systems on this as well.

Finally, several conclusions can be drawn from the mapping of our software and system evolutions. These will be summarized in the final section of this paper.

2. Patterns of Technology Evolution: a Model

In [3, 4] a model of technology evolution is proposed based on the concept of a *technology ecosystem*. The authors made the model based on the insight that the evolution of a technology cannot be understood by studying one technology in splendid isolation: technology innovates and evolves by interaction with the innovation and evolution of other technologies. In a natural ecosystem, the evolution of one species is influenced by the evolution of many other species. And similarly the evolution of the species being studied influences the evolution of many species in its environment, which may in the end result in influences that drive the evolution of the species under study again. The authors of [3, 4] based their theory on the fact that similar patterns apply to innovation and technology evolution as well.

In the technology ecosystem theory, three roles are identified that technology can play in a technology ecosystem:

1. The *component role*: this role is played by technologies that are used as components in more complex technologies.
2. The *product and application role*: this role is played by technologies that use components to perform a set of functions or satisfy a set of needs. They are defined by the components they use and the services they provide.
3. The *support and infrastructure role*: this role is played by technologies that work in conjunction or collaboration with, or as a peripheral to other technologies.

The difference between the component role and the support and infrastructure role is that components are necessary for the design, and are part of the physical structure of a technology, while support and infrastructure technologies simply work in combination with other technologies (definitions taken from [3,4]).

Technologies playing different roles influence each other's evolution in different ways as depicted in the figure below (taken from [3]):

- Component evolution enables product evolution by designing new or improved products that exploit the value of component innovations.
- Product innovations create a need for innovations in components and influence the evolution of component technologies this way.



- Product innovations are being adopted in the market and create the opportunity for developing new services based on these new products (e.g., the adoption of digital cameras resulted in internet services for printing digital photos and creating digital albums)
- Innovation of services create a need for innovations in products and components (e.g., the ability to share pictures on-line results in new standards for the external interfaces of digital cameras)

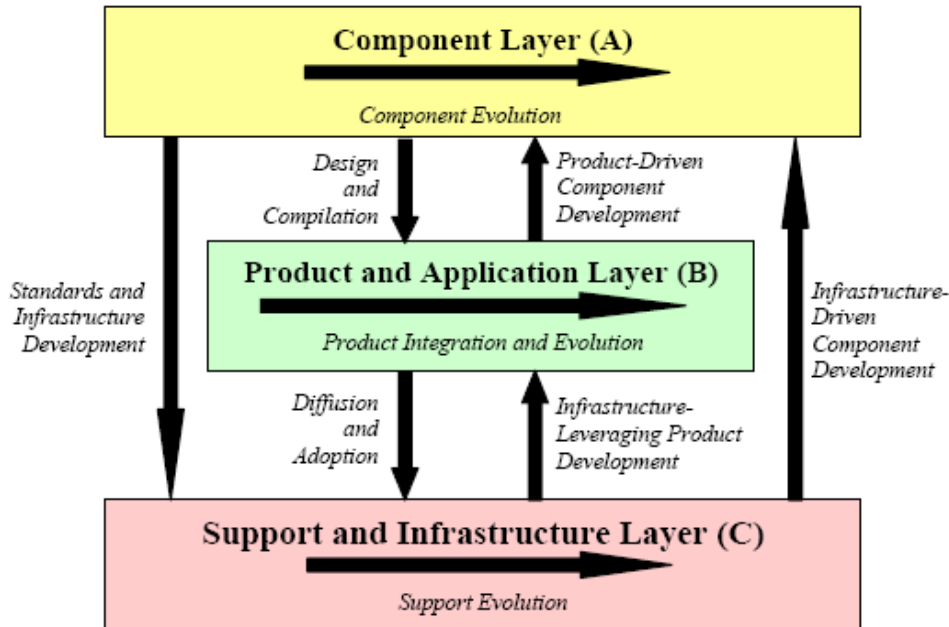


Figure 1: Paths of Influence between Technologies (from [3])

To analyse the evolution patterns of the software embedded in our systems, we interpreted the model as follows:

- When the evolution of our products is primarily autonomous (i.e., innovations stay in the product innovation layer and are not driven by innovations in the component and support/infrastructure layer), the software in our systems can also have a rather autonomous evolution pattern.
- If, however, the innovation of our system is to a large extent driven from the components and support/infrastructure layer, the evolution of the embedded software can hardly be considered autonomous.

In the latter case, planning of evolutionary software development has to be well-aligned with the evolution in the other technology layers. In that case, an integrated approach for software evolution and long-term product innovation planning (roadmapping) has to be found.

In the paper of Adomavicius et al, the model is applied to chips and MP3-technology. We will apply the model to our medical equipment in the next section. We will introduce the way the relations amongst technologies are visualised as we discuss the evolution of our systems in the next section.



3. Patterns of Technology Evolution: Four product lines in PMS

In Philips Medical Systems, a large range of medical equipment is being developed. Most of these products are large, complex systems with several millions lines of software code. To study the evolution patterns of these systems, we applied the model discussed in the previous section. We will provide four pictures depicting the influences between the innovations in the three roles (C = component; P = Product/Application; I = Infrastructure/Support). One (the X-Ray equipment for the cathlab) we will discuss in some details (without giving away our plans for future innovation on these system). The other three we will only discuss briefly. For these we will only provide an overview of the result of the analysis.



Figure 2: The X-Ray Equipment in a Cathlab

Evolution of the X-Ray Equipment for Cathlabs.

For treatment of heart and vessel diseases, X-Ray equipment is used in the cathlabs. By means of X-Ray imaging, this type of equipment allows the cardiologist and radiologist to view the heart and to position catheters in the heart. These systems (see figure 2) consist of (i) a patient table for carrying the patient, (ii) one or two C-arcs which carry the X-Ray tube(s) and the X-Ray detector(s), (iii) computer hardware and software for processing the acquired images and for planning the procedure, (iv) a series of displays mounted next to the patient table and in a separate control room. The cardiologist and/or radiologist will be standing next to the patient where he will insert catheters into the patient's arteries (typically in the groin). Using the visual feedback he gets from the X-Ray images, he will position the catheter. One of the distinguishing features of these systems is the latency of the images, since a very short latency is essential for the eye-hand coordination of the doctor.



For the period of 2000-2010, we analysed the major (expected) innovations. We identified six innovation steps in this period (see figure 3):

- **Step 1:** $(C \rightarrow C^*)$ ¹
New component technologies allow the development of new flat X-Ray Detectors. Initially, these detectors are put on new systems by replacing the at that time current X-Ray detectors. The systems are not fully optimised to leverage the new features of these detectors.
- **Step 2:** $(C \rightarrow P^*)$ and some $(P \rightarrow P^*)$
The new X-Ray detectors are being incorporated in new systems and systems are being designed to exploit the benefits of these new detectors and to offer additional value to the clinical users. This results in improved image quality and improved ease-of-use.

At the same time, the architecture of the system is changed significantly. This was triggered by the event of the new detector technology, but not fully dedicated to this innovation. As a consequence also other product innovations are resulting from this such as improved ease-of-use, better system interoperability etc.

- **Step 3:** $(C \rightarrow P^*)$ and $(P \rightarrow P^*)$
In the area of X-Ray detectors, new variants of the initial detector emerge with more pixels and a larger imaging area. These detectors were designed for a different application domain (vascular procedures rather than cardiac procedures for which the initial version was designed). In combination with new components for image processing, the characteristics of this new vascular detector can be exploited by providing a $2K^2$ (= 4 megapixels) imaging chain for vascular procedures. This used to be a $1K^2$ (= 1 megapixel imaging chain) of 512^2 (=0.25 megapixels) imaging chain.

New image processing technology is also used to improve image quality by adding much more effective image processing algorithms to the system. This results in improved clinical outcome by providing better images or in reduced X-Ray dose for the patient and for the clinical staff by using the enhanced noise-reduction capabilities of the image processor.

The $P \rightarrow P^*$ results from a series of functional extensions of the system that are neither based on innovative components, nor triggered by innovations of infrastructure technology.

- **Step 4:** $(P \rightarrow P^*)$ and $(I \rightarrow P^*)$
The next step consists of a series of functional extensions of the systems without any real major contribution from new components. Innovative components that are being developed in parallel did not result in major innovations of the X-ray products in that period

¹ This notation has been taken from [Adomavicius2005]: $A \rightarrow B^*$ means that an innovation in domain A results in an innovation in domain B. B^* refers to the new state in domain B. $C \rightarrow C^*$ therefore means that an innovation in a component technology results in new component technology.



In these years, 3rd parties developed innovative products that were to be used in the Cathlab in conjunction with the X-Ray system. Initially, these systems were stand-alone systems. But as the consequence of their stand-alone nature, the workflow in the Cathlab was not optimal. The interoperability between the X-Ray systems and the new equipment was low and the customer value was not optimal.

Step-by-step, the external interfaces of the system were extended to provide optimised workflow in the Cathlab again.

- **Step 5:** ($I \rightarrow P^*$) and some ($C \rightarrow C^*$)

In this step we are looking forward into the future. This means that we cannot be very explicit about the innovations in this step (to avoid putting confidential information in the public domain).

In short, we foresee that innovation in component technologies will allow us to optimise the system architecture of our Cathlab systems again. Furthermore, a continuous stream of 3rd party innovations will allow us to enhance our products.

- **Step 6:** ($C \rightarrow C^*$) and ($C \rightarrow P^*$) and ($I \rightarrow P^*$)

In addition to yet another round of 3rd party integration opportunities, we expect component technologies to result in components with significantly improved characteristics, which will allow us to innovate our products.

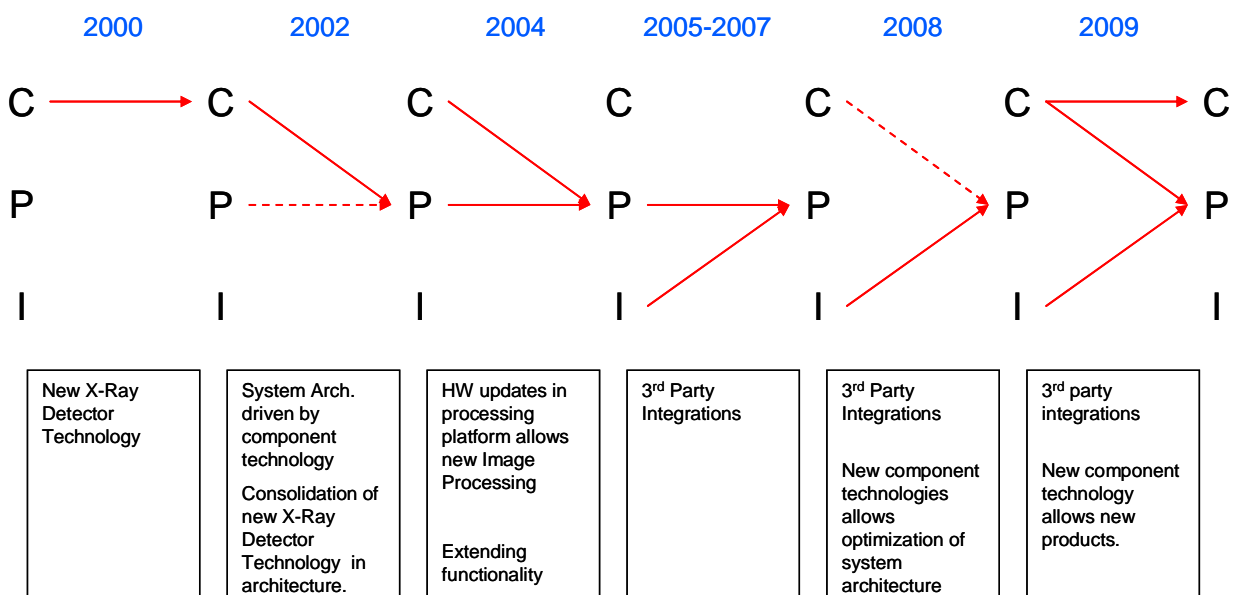


Figure 3: Innovation and Evolution Map for Cathlabs

In all of these steps, we could have added the ($P \rightarrow P^*$) arrow, since we see a continuous stream of evolutionary improvements of our systems: (i) adding new features, (ii) resolving field problems, (iii) removing customer dissatisfiers, (iv) optimising existing functions, (v) cost-price reduction, etc. Similarly, each version of a component will have improved characteristics, due to small improvements of component technologies ($C \rightarrow C^*$). And similarly, a continuous stream of innovations takes place in the system's environment ($I \rightarrow I^*$), but most of these innovations did not



have impact on the X-Ray system. In our drawing and in the corresponding brief chronicles of almost 10 years of innovation in X-Ray Cathlabs, we focussed on those innovations that had significant impact on the X-Ray system, since we wanted to study the drivers for evolution of the software in these systems.

Evolution of other types of Medical Equipment.

Next to creating the innovation map for the X-Ray equipment in Cathlabs, we conducted a similar study for three other product lines in Philips Medical Systems. Without a detailed explanation of all individual arrows in these pictures, the resulting maps are shown in figure 4-6. In the MRI-map, we see a strong drive from component technology innovations to product innovations. Recently, we also see a strong drive from infrastructure innovations. As in the Cathlab, this is due to the integration of innovative 3rd party technologies.

In the IP-Map (figure 5) we see a strong drive from component technologies to product innovations. In this map, the drive from infrastructure technology is absent. In the innovation map for the software component platform (software) we see a much more product-driven innovation. Component technologies are very important (java, C#, .Net. Windows XP, etc.) but we also see a stream of innovations in the platform that is to a large extent independent from innovation in component technologies. These innovations are primarily functional extensions of the platform and improvement of software quality attributes (performans, reliability, etc.) The (I-P*) arrows we see in this map are caused by innovations with respect to security standards and interoperability standards.

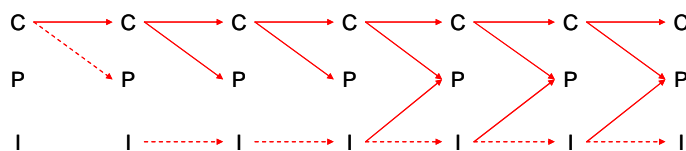


Figure 4: Innovation and Evolution MRI Scanners

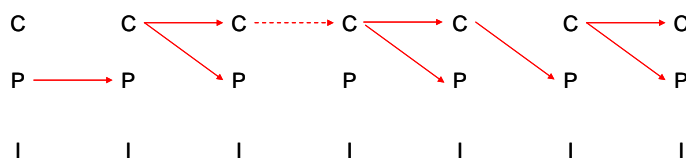


Figure 5: Innovation and Evolution Map for Image Processors

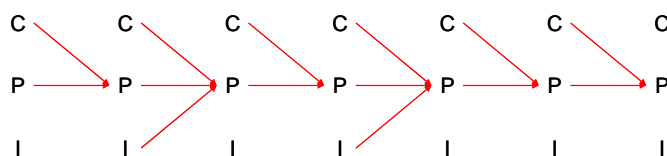


Figure 6: Innovation and Evolution Map for the Software Component Platform

When looking at the evolution maps we sketched for the four product lines, we concluded that only a small portion of product innovations are in control of the development organization. Many innovations are driven by component innovations.



In more recent years and in the near future we also see a major influence from “infrastructure technologies” developed by 3rd parties. Integration with these “peripheral systems” will make important contributions to the innovation of medical equipment.

With respect to software evolution we concluded two things after drawing these maps:

1. Since the evolution of external technologies ($C \rightarrow P^*$) and the evolution of infrastructure technologies ($I \rightarrow P^*$) have a major influence on product evolution, software evolution most probably cannot be seen as an autonomous process. Software evolution takes place in the context of product evolution, which is subject to strong external influences. This means that the evolution of software needs to be guided by a component and infrastructure technology roadmaps; a longer-term planning process is needed to make sure that the software evolves into “the right direction”.

We also noted that the nature of the evolution of the software components platform (figure 6) has stronger ($P \rightarrow P^*$) evolutions. Although component technology innovations are also relevant for the evolution of the software platform, the software appears to have a much stronger autonomous evolution pattern. This software has a much more generic nature (not specific for a certain system or application domain). This may very well explain why it has a different evolution pattern.

2. So far, the software evolution is considered part of the product evolution. But product evolution has many technology aspects. What is the relation amongst these?

In the next section we will look into the product technology innovation in a little bit more detail.

4. Patterns of Technology Evolution: putting Software on the Map

After studying the evolution patterns of four product lines in Philips Medical systems, we concluded that we would need to look into product innovations in more detail. Software is only part of our systems. In some cases, innovations are fully driven by software. But in many cases, software is only part of the innovation. And in some cases, software only enables the innovative features of new components (in that case, the software itself only makes a marginal contribution to the innovations).

The products we analysed have a large hardware component. What’s more, these hardware components drive to a large extent the value of the system. If we look at the X-Ray equipment in the cathlab, it is clear that a major part of the cost and value of the system is determined by hardware components: X-Ray tubes and High-Voltage generators, X-Ray Detectors, the Image Processing Hardware, the large mechanical components for carrying and positioning the patient and the X-Ray Tube/Detector etc. The same applies to the MRI-scanners. In these systems, software starts to play a dominant role in unlocking the value of the hardware components ($C \rightarrow P^*$) innovations and by providing optimal ease of use etc. ($P \rightarrow P^*$, but also many $I \rightarrow P^*$ innovations: integration with the hospital IT-systems). As noted at the end of the previous section, the evolution pattern of the software component platform has a different nature: stronger $P \rightarrow P^*$ innovations and therefore more autonomous evolution steps.



To study the evolution patterns of software in medical equipment, we distinguished three technology aspects in our systems:

- Software
- Hardware
- System Architecture (which defines how all hardware and software components combined into one system)

We asked ourselves the following question: do innovations start in the software or do they start in hardware and system architecture innovations?

To answer this question, we took the previous evolution maps and we split the Product evolution into the three aspects listed above (SA = System Architecture, HW = hardware, SW = Software). For the Cathlab, the result is shown in figure 7.

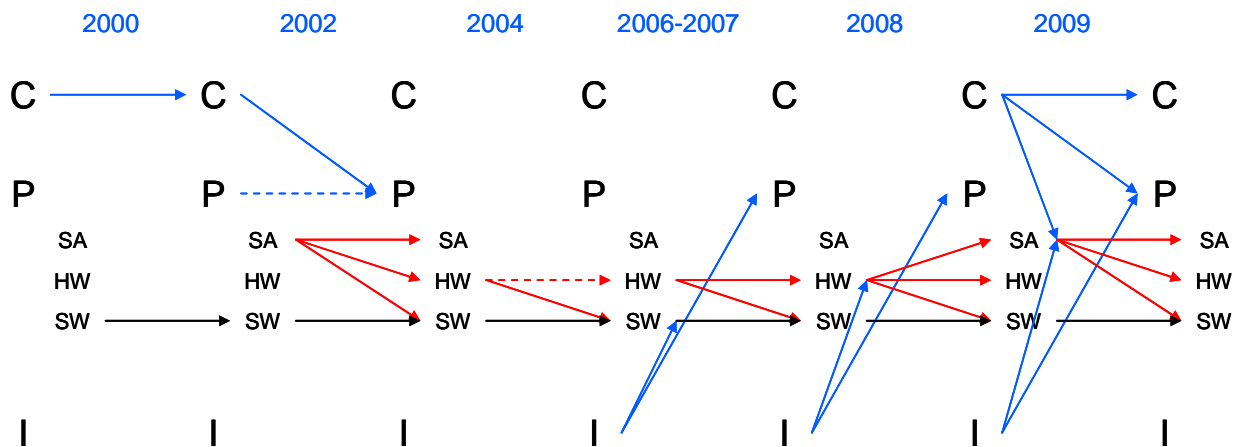


Figure 7: The Extended Cathlab Evolution Map: HW, SW and System Architecture Added

In order not to obfuscate the picture we did not highlight the SW→SW* arrows. As can be seen, they do exist: functional extensions, corrective maintenance, workflow optimisations etc.

But, as can also be seen: most of the major innovative changes are driven either from hardware technology innovations or from the changes to system architecture. The system architecture itself changes due to: (i) component technology innovations facilitate a different (more effective) system architecture or (ii) infrastructure technology innovations require the system architecture to be changed in order to make integration with innovative 3rd-party products and technologies easier.

We have also made similar pictures of the software evolution patterns for the MRI-scanner and for the Image Processors. For these we see the same patterns: in addition to evolutionary changes to the software, changes to the system architecture and to the hardware of the product have a major influence on the evolution of the software. And in addition to that – as in Cathlabs – most major innovations start in component and infrastructure technologies.



5. Conclusions and Future Work

We started this work as a first step of our contribution to the ITEA Serious project: define an approach to make evolutionary software development fit with the longer-term planning processes (roadmapping) that are needed to optimise the investments in innovation of our medical equipment. This approach would have to take into account: (i) the long operational period of our systems (10+ years operational in the field), (ii) the nature of our software: embedded in complex, capital-intensive systems and (iii) the fact that many major innovations in our system do not start in software technology (some have, but many have and will not).

Our hypothesis was that many factors that are outside the direct scope of software evolution have a major impact on the software evolution. We assumed that the influence of these factors could result in software revolutions rather than software evolutions. To study this phenomenon, we started to study evolution patterns of our products and of the embedded software. We wanted to clarify which factors influence the software evolution. We therefore sketched the evolution patterns of four product lines, using the technology ecosystems theory of [3, 4].

From the resulting sketches of the evolution patterns we concluded that:

1. Component and infrastructure technology evolution does indeed have a major impact on product/software evolution.
2. We also noticed that the software-only product that was part of our study (the software component platform) “suffers” far less from this phenomenon. For this type of products, the evolutionary approach is likely to be a much easier fit.
3. Many major innovations start with system architecture and hardware innovations. In many cases, the software evolution follows evolutions in other domains.
4. There is a continuous stream of evolutionary innovations in all system components: also in software. For these, an evolutionary approach would be very well suited.

To optimise evolutionary development in software (for the continuous stream of evolutionary innovations) we have to define a proper way of integrating out longer-term (roadmapping) processes with the evolutionary approach. Our innovation maps, depicting typical innovation patterns will serve as a guideline. The next step of our work will be to study our current roadmapping and development processes, and to collect our best practices (and those of other partners in the ITEA Serious consortium). Based on these we will propose a roadmapping approach that allows us to optimally leverage the benefits of evolutionary software development.

6. Literature

- [1] Tom Gilb, *Principles of Software Engineering Management*, Addison-Wesley Publishing Company, 1988
- [2] <http://agilemanifesto.org/>
- [3] Gediminas Adomavicius, Jesse C. Bockstedt, Alok Gupta, and Robert Kauffman, *Technology Roles in an Ecosystem Model of Technology Innovation*, March 3, 2005
http://www.misrc.umn.edu/workingpapers/fullPapers/2005/0504_030305.pdf



[4]

Gediminas Adomavicius, Jesse BockStedt, Alok Gupta, and Robert Kauffman, ***Understanding Patterns of Technology Evolution: An Ecosystem Perspective***, Proceedings of the 39th Hawaii International Conference on System Sciences, HICSS, IEEE Computer Society, 2006
<http://doi.ieeecomputersociety.org/10.1109/HICSS.2006.515>

Acknowledgements

Thanks to Peter van der Meulen of PMS MR for participating in the study.

This paper is written as part of Task T2.3 of the ITEA Serious project. It corresponds to section Y.1-Y.3 of the planned deliverables.

