

Evaluation of the Architecting Method

-

1. product is a commercial success	
+ sales volume + selling price + margin + time to market	✓ derived from Figure 16.3
2. product family is sustainable commercially successful	
+ 3 products + 10 releases in 3 years	✓ derived from Figure 14.10
3. architects benefit from deploying submethods in multi-view framework	
submethods CA ? F ✓ CR ✓	derived from sections 18.4.1 18.4.2 18.4.3
integration of the method	derived from sections 18.4.4 18.4.5
qualifies checklist	✓
story telling	✓
multi-view framework	✓
reasoning	✓
4. project leaders, product managers and engineers are able to use the outcome of the submethods	
results used by stakeholders for many purposes	✓ too late ? derived from Figure 18.7
	too abstract

Legend: CR CR ✓ CR

Gerrit Muller

Embedded Systems Institute

Den Dolech 2 (Laplace Building 0.10) P.O. Box 513, 5600 MB Eindhoven The Netherlands

gerrit.muller@embeddedsystems.nl

Abstract

The case study is evaluated: the resulting product and its design and the way the method has been used by the product creation team. The evaluation is done by means of the predefined hypothesis and criteria.

Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

All Gaudí documents are available at:
<http://www.gaudisite.nl/>

version: 0.9

status: finished

February 10, 2011

1 Introduction

The hypothesis formulated in Chapter ?? is evaluated by means of the Medical Imaging Workstation case. This evaluation uses the criteria that have been defined in Chapter ??.

Figure 1 shows how the case maps on the hypothesis and repeats the criteria. This figure is used as the basis for the evaluation of the case.

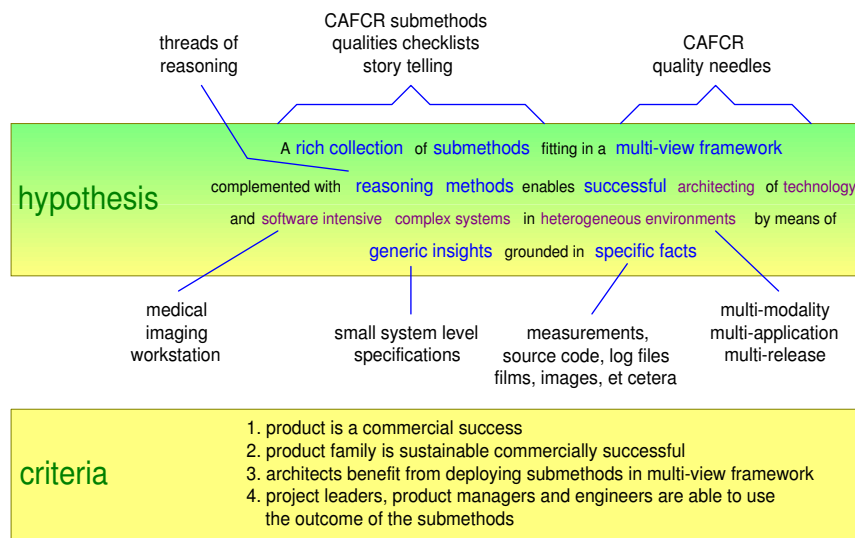


Figure 1: Annotated hypothesis and criteria as basis for the evaluation

The hypothesis narrows the scope: the method is applied on *software and technology intensive complex systems* and the product creation takes place in *heterogeneous environments*. The medical imaging workstation fully fits in these constraints. The system requires many hardware and software technologies, see Figure ?? and the list of technology innovations in Section ?. Also a lot of physics and clinical application related technologies are required to translate pixels into clinically relevant information. The heterogeneity of the environment is present in multiple dimensions: *modalities* (X-ray, CT, MRI), *applications* (gastro-intestinal, vascular, CT and many more) and *releases* (URF, vascular, CT, and MRI; each of them interoperable with multiple releases).

Section 2 evaluates the design of the product. The design quality is a measure for the sustainability of the product and for the support of the method to the PCP team. Section 3 evaluates the product itself. The product evaluation addresses the criteria 1 and 2, of Figure 1, which address the *commercial* success. Section 4 discusses how the *method* has been instrumental in creating the product and how it has contributed to sustainable success, thus evaluating criterion 3: Criterion 4, the

usability of the outcome, is evaluated in Section 5. The conclusion of the complete evaluation is articulated in Section 6.

2 Design Evaluation

Conceptual		Realization
+ notification	+ processing pipeline	+ memory management
+ Objective-C	+ graphics	+ DB based communication
+ standard workstation	+ UI toolbox	+ SW keys
+ X bypass	+ PMSnet	+ OIT
+ Unix	+ database engine	
~ modularity		
~ distance internal and external information model		
~ some bloating due to over-genericity		
~ property handling		
- dependency structure		
- interface management		
lots of discussions about : language choice (why not C++) windowing system platform re-use		

legend

+ good

~ doubt

- problem

based upon technology assessment in "Technology Improvement Plan"

Figure 2: Evaluation of the design of the medical imaging workstation

Figure 2 shows an evaluation of the design quality, which is based on an internal technology improvement plan [2].

The community of designers was very content about many of the concepts and implementations. This assessment is rather subjective, because the designers are creators and users of the design at the same time. The following three questions are a somewhat more objective way to make the quality assessment more explicit:

- Does the product, based on this design, fulfill the requirements?
- How does the design support ongoing developments? How is development effort (in time and manpower) affected when new functionality is added, for instance as shown in Figure ???
- Benchmarking: How does the product and design compare with other products? For example, compare code size in relation to the offered functionality for multiple products.

Most of the success of the first product and its successors, as described in Section 3, are due to the power, flexibility, expressiveness, and richness of the concepts and implementations mentioned in the “good”-section of Figure 2.

Note that most of the strong points are concepts that have been chosen and implemented by the software designers rather than by the architect. The SW designers deserve the credit for the selection and implementation of these concepts. The architect is involved in the selection and deployment of the concepts, but more supportive and monitoring than leading. The submethods that have been presented do not deal with the selection of these concepts, although they provide some means such as criteria. The submethods mostly provide means at higher integration levels.

A *code size comparison* is hard to reconstruct at this moment, mostly because it requires an objective measure of functionality. The 360kloc (including comments) of code in the medical imaging workstation divides about equally into code for viewing functionality and code for the modality products. However the workstation removed many of the constraints of the modality products. It allowed, for example, arbitrary sized images instead of 512^2 and 1024^2 images only. In other words more functionality is offered in about the same amount of code.

The designers who actually used the concepts and the implementation were very positive. People outside of the medical imaging workstation product group, however, had a lot of critical remarks about the design and implementation. The feedback of this group was not based on practical experience but rather from documentation (best case) or hearsay (worst case). This situation was caused by the political climate within the division, where competition for funding caused a lot of this kind of non-rational arguments.

The critical remarks of the group of outsiders were taken seriously. As a result a technology improvement plan was written, on which figure 2 is based. The aspects assessed as “doubtful” and negative are discussed below.

The first phases of the improvement plan focused on improvements of the *modularity*, *interface management*, *dependency structure*, and the *distance between internal and external information model*. The problem with modularity was that only fine-grain modularity, at class or file level, existed. No higher aggregation level existed and no explicit dependency structure was defined. In general the class level granularity was good. Some classes, however, were too generic, which resulted in bloating. Inheritance was used too much, which often happens when development teams start to use OO techniques.

The internal and external data models were two entirely different entities managed by different people. From an interoperability point of view this is a risk. This risk becomes more prominent when different modalities have to interoperate.

The delayed design choices, and data about installation, configuration, and customization were all stored in property files. This mechanism did not scale up well and the information was not recognized as regular code. The risk is comparable with missing source code management. Another risk is loss of overview. Note that those risks were not yet acute in the first product with its limited configurability and functionality.

The interface management was one of the main issues of this design. The

product did not suffer from any interface management problem, but follow-on products, which re-used the code base, complained a lot. All changes applied somewhere in the code could propagate to other parts of the code.

In the organization a lot of noise was present in discussions about the programming language (Objective-C [1]). The opinion of managers and engineers outside of the project was that Objective-C was a non-standard language, which was dying due to lack of followers in the rest of the world. On the other hand, a lot of the valuable concepts built upon the dynamic nature of Objective-C. These concepts would have bloated significantly if C++ would have been used for the implementation. In retrospect, a language disappears not as fast as predicted by many of its opponents. Somewhat less heated was the debate about the windowing system. Most organizational noise was removed by the use of the X-compatible bypass (Nix).

A lot of disturbance was caused by the platform expectations in the organization. The original objective of the development team [4] was to create a reusable platform for Philips Medical Systems. The team focus changed to create medical imaging workstation products. From the product management point of view the platform creation became a non-issue: platforms are not sold to external customers! The product groups in the context did not follow this focus change. They expected a cost reduction to be achieved by sharing the development costs of a viewing platform.

3 Product Evaluation

The case describes the development of the first product release of the Medical Imaging Workstation. Figure 3 shows the strong and weak points, as they were perceived at the outside of the product.

The feedback from the URF customers, obtained via application managers who visit many URF departments, was very positive about the *usability* of the *film layout* and the *film efficiency*. Also the *operator efficiency of printing* and *ease of auto-printing* was appreciated. Viewing was used only very limited. The few users who used it also for viewing complained about the simultaneous performance of viewing and auto-printing. Note that this was no surprise; it was even according to specification. It indicates that to a small subset of the customers the specification was insufficient.

At specification level the *throughput*, *image quality*, and *interoperability with URF systems* were according to specification. The specification is appreciated by the customer. The interoperability with vascular systems was below internal (product management) expectations. Product management expected vascular systems to interoperate in the same way as URF systems. However, vascular systems are used quite differently than URF systems. The functionality and performance needs

	C ustomer objectives	A pplication	F unctional
customer feedback	++ usability film layout ++ film efficiency + operator efficiency printing + ease of auto-printing - concurrent viewing and auto-printing		+ throughput + image quality + interoperability URF - interoperability vascular
operational feedback	+ sales volume + selling price + margin + time to market - return on investment		+ manufacturability + option handling ~ network installation

legend

- + good or
- ++ very good
- ~ doubt
- problem

Figure 3: Evaluation of the medical imaging workstation product

of vascular surgeons differ from the radiologist's needs. The system did not provide significant value for vascular use. The vascular product group did not have a clear incentive to improve the interoperability, because of the lack of added value, Not many customer complaints were filed about this problem¹.

From a business point of view the product was clearly a success. It yielded a good *sales volume* (increasing from circa 50 systems in the first year to hundreds of systems in later years), the right *selling price*, a reasonable *margin*, and a good *time to market*. The return on investment (ROI)², however, only occurred after several years.

At the portfolio level of Philips Medical Systems (PMS) the turnover of these workstations was negligible. Modality systems (MRI, X-ray, CT-scanners) form the bulk of the turnover. Modality systems have a limited innovation speed, due to their multi-technology complexity and their safety requirements. The medical imaging workstation provided a means to introduce *image handling* innovation faster, because this product has its own (more dynamic) life-cycle. In later years the strategic value of an independent platform for image handling innovation at PMS level turned out to be large. Additional modality³ turnover could be realized thanks to the image handling capabilities of the medical imaging workstation.

¹Service engineers and application managers file complaints of users by means of Problem Reports. A control board analyses and discusses incoming Problem Reports regularly.

² Product groups have a profit/loss responsibility. Losses are accepted for about 2 years, after such a start-up period a "normal" ROI is demanded. The volume of all products, including options, was after about 2 years at an acceptable level. Especially the (software) options helped to get at an acceptable ROI.

³The term *modality* system is used for image acquisition systems. The image acquisition technique, such as MR, CT, US, or X-ray, is dominant in the system design. This technique is called the modality.

Feedback from manufacturing and service departments indicates that the operational requirements, such as *manufacturability* and *option handling*, were fulfilled satisfactorily. Installation of networking functionality overlapped with other hospital disciplines, such as IT departments and facility management. For the Philips service work force this was a new area. The networking was not a big issue, but it required quite a lot operational attention.

4 Evaluation of Architecting Method

Subsection 4.1 evaluates the use of the submethods, described in the Chapters ?? and ?? and applied in the Chapters ?? and ??. The use of the qualities and the quality checklist (Chapter ??) are evaluated in Subsection 4.2. Subsection 4.3 evaluates the story telling (Chapter ?? and ??). The reasoning methods (described in Chapter ?? and applied in Chapter ??) are evaluated in Subsection 4.4. The integration of all components of the methods is evaluated in Subsection 4.5.

4.1 CAFCR Submethods

Figure 4 shows the submethods per view. Every method is annotated with its usage in the case. Many submethods are used explicitly, although not always exactly as described in part II. The fact that these submethods have been used recognizes the need for these methods.

Some subjects covered by submethods have been discussed during the product creation, but did not result in any specific consolidation of data in specifications. In figure 4 these submethods are labelled “*only implicitly*”. Often the coverage of these subjects was rather partial. They have been discussed, but they were not thoroughly analyzed.

The fact, for instance, that safety has not been explicitly addressed in the first release is due to the low severity factor of this product. This product does not immediately endanger patients or operators. X-ray systems, on the other hand, use radiation and heavy moving parts that can be dangerous for patients as well as personnel. These systems take measures to cope with the risks, such as shielding and detection. In later releases of the medical imaging workstation the safety has been addressed explicitly, with a mandatory hazard analysis and accompanying specification and design measures.

The subjects that have not been covered in the first release were less important and critical than the covered subjects. This selection process is described in Figure ??. Some of these subjects, for instance security, have been addressed explicitly in a later release.

The most remarkable observation from Figure 4 is that the *Customer Objectives* and *Application* views are poorly covered. As discussed in Section ?? the introvert culture of the development teams causes this unbalance.

C ustomer objectives	A pplication	F unctional	C onceptual	R ealization
<p>key drivers value chain</p> <p>business models suppliers</p>	<p>context diagram</p> <p>stakeholders and concerns</p> <p>entity relationship models dynamic models</p>	<p>case descriptions commercial decomposition service decomposition goods flow decomposition function and feature specifications performance external interfaces standards</p>	<p>construction decomposition functional decomposition designing with multiple decompositions execution architecture internal interfaces performance start up shutdown integration plan</p> <p>work breakdown safety</p> <p>reliability security</p>	<p>budget benchmarking performance analysis granularity determination</p> <p>value and cost</p> <p>safety analysis reliability analysis security analysis</p>
legend	explicitly addressed	addressed only implicitly	not addressed	

coverage based on documentation status of first product release

Figure 4: Coverage of submethods discussed in part II

4.2 Qualities

Figure 5 shows the use of qualities that appear in in the documentation structure in 1996, when the product family was already more mature. A significant amount of qualities were explicitly documented in separate documents: *image quality*, *safety*, *security*, *throughput or productivity*, *connectivity*, *resource utilization*, *configurability* and *installability*. Many other qualities were addressed as part of other documentation, such as functional requirement specifications or the technical product documentation.

Figure 6 shows the relative coverage of the qualities as they have been applied in the case. This coverage is determined by the amount of explicit and implicit information present in the system documentation, see Figure 5. The relative relevance of these qualities for this business is also shown. The relevance is based on experience used in a retrospective assessment. The qualities in this figure have been aggregated. This aggregated format makes it immediately clear that only a subset of qualities is relevant per business.

For medical imaging qualities such as *ecological*, *logistics friendly* and *down to earth attributes* are relatively unimportant. In other systems, for instance printers, all of these qualities are important.

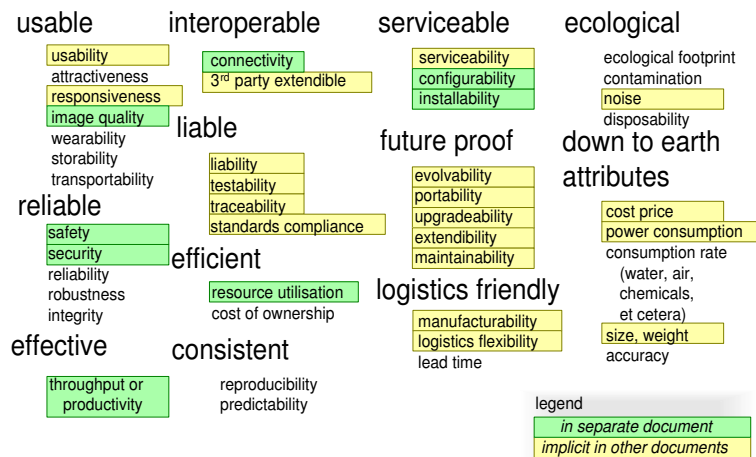


Figure 5: Quality documentation in 1996

The conclusion of Figures 6 and 5 is that the qualities play an important role in the product creation. For in this product many qualities were even documented in separate documents.

4.3 Story Telling

Story telling has not been deployed in the medical imaging case in the way described in in Chapter ???. In the documentation of the system there is a document called *typical case*. This document partially fulfilled this role. The *typical case* in combination with the oral version, described in Chapter ??, proved to be very valuable to connect the clinical world and the engineering world. The radiologist story created the focus on printing on film. The efficient film production became the number one selling argument of the product.

4.4 Threads of Reasoning

The threads of reasoning were only semi-consciously applied. The fast iteration over multiple views was consciously applied. The integral understanding emerged as a result of this fast iteration. In retrospect this approach can be visualized as a *thread of reasoning* as described in Chapter ???. It is this integral understanding that helped to finish the product within the constraints. The integral understanding can indirectly be observed by the handling of the problems found during integration, see Sections ?? and ??.

The introduction date is a very important business constraint: late introduction means lower market share and lower margins. The integral understanding made it possible to finish the product in time. Without integral understanding it would have

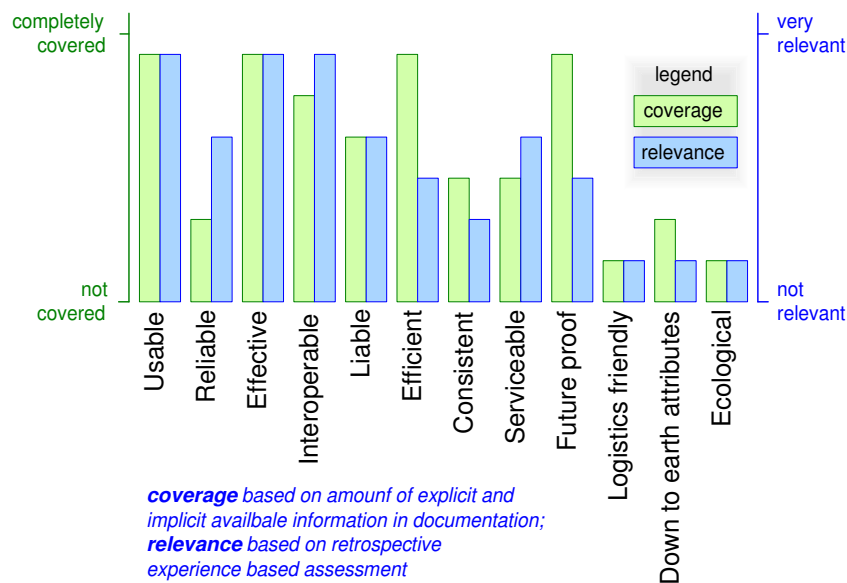


Figure 6: Coverage profile of qualities

taken much more time (months or years) to get the product at an acceptable level.

4.5 The Integration of the Method

The hypothesis starts with “A rich collection of submethods fitting in a multi-view framework complemented with reasoning methods”. In other words the *submethods*, the *multi-view framework* and the *reasoning methods* must be employable in a complementary fashion. The integration of components of the methods is evaluated in this subsection.

The main value provided by the architecting method is in detecting and solving **integral** problems, such as the *throughput*, *image quality*, and *memory utilization*. Most local aspects (single concepts, single functions) were engineered in a professional way. Integration problems arise across the boundaries of concepts and functions, and emerge due to the dynamic behavior of many components. The submethods and the framework add means to do the *system* level design.

Many submethods are based on very specific and detailed facts. The *typical case*, as described in Section ??, also focuses on a completely quantified description, which enables specific analysis and design actions. The specifications at the system level have to be rather generic to keep the specifications usable and the size manageable. Typical quality specifications, for instance, were between 4 and 8 pages of content. In other words, in the consolidation of the design there is very limited room for specific details.

The architect continuously operates in this field of force: the need for *genericity* in guidelines and rules, while the added value is mostly in the integral understanding emerging from a large amount of highly *specific detailed facts*.

In the medical imaging workstation case the amount of viewpoints, submethods, and qualities used was good: no dramatic problems caused by missing viewpoints, submethods or qualities arose later. All of the viewpoints, submethods, and qualities are highly relevant in solving the integration problems. The duration of the integration of the first product and of the later products, see Figure ??, was manageable. The relevance of the viewpoints, submethods, and qualities is shown in Chapter ?. The amount of method components was manageable for the architects, and the results could be absorbed by the stakeholders. The selection of the methods emerged by (unconsciously) applying the threads of reasoning. Conclusion: the reasoning method resulted in a minimal set of required components of the method, and the reasoning method supported the integration of these components.

The balance between genericity and specificity, which is formulated at the end of the hypothesis “*by means of generic insights grounded in specific facts*”, did work out well in the case. Sufficient specific details, such as measurements, source code, log files, films, and images, were touched to form the basis for the emerging understanding. The size of the system level documentation was rather limited; the integral overview was clearly present in the documentation, the overview was not hidden in a vast amount of details. Some of the engineers, however, criticized the abstraction level of the documentation. This criticism is discussed in Section 5, which evaluates the usability.

5 Usability Evaluation of the Outcome of the Architecting Method

Criterion 4 states “*project leaders, product managers and engineers are able to use the outcome of the submethods*”. This section evaluates how the PCP team members have experienced the outcome of the submethods.

The results of the submethods in the CAFCR views are consolidated in product and design specifications. Figure 7 shows who uses the results, what the results are used for, and what complaints have been voiced by the engineers. These specifications have been used by a wide variety of stakeholders: *product management, application, project leaders, engineers, test engineers, purchasing, manufacturing, and suppliers* for a wide range of purposes: to *create detailed specifications*, to *test*, to *communicate*, to *derive documentation* such as manuals, and as basis for *succeeding products*.

Some engineers complained about the level of abstraction of the output. “How can I use this specification; I need more guidance”. The problem is that the distance from system level specifications (high abstraction level) to module level implemen-

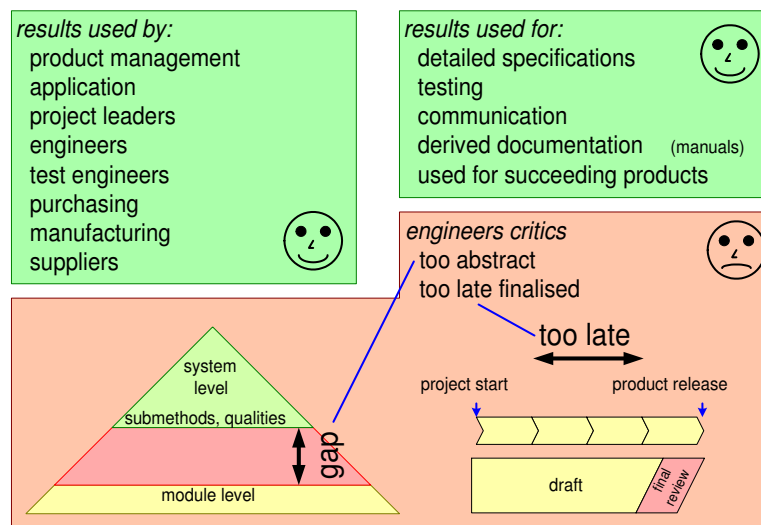


Figure 7: Users and usage of the results of the architecting method

tation (low abstraction level) is very large (from tens of documents to hundreds of thousands lines of code). For some functional subsystems this gap was bridged by deploying the architecting method recursively. Some members of the product creation team were capable of doing this, but for other subsystems the gap never got bridged.

The solution for this class of problems is often outside the scope of the method. This gap problem, for instance, can be solved by working on the quality of the PCP team. To improve the team quality the skills of the individuals can be improved and the composition of the team can be changed to obtain the required distribution of skills in the team. The skills of the designers at the module level are increased to be able to work directly from the system level output. Nevertheless, this will always be an area of tension: problems that are complex by nature cannot be simplified infinitely. The quality of the PCP team is solved in the wider context of the PCP, as shown in Figure ??: people (skills), organization (team composition), process (methods).

Another complaint uttered by some of the engineers was that the results were frozen quite late. Many specifications exist for a long time in *draft* status, to get their final review somewhere in the integration phase. On the one hand this is a fair criticism: most engineers need stable information to make good progress. The problem on the other hand is that product creation in complex and innovative environments is full of uncertainties: is it feasible?, what do we need exactly? how much does it cost? And these questions are often mutually dependent. No method will ever be able to know the unknowns; in the best case it will help to cope with the unknowns. Support for unknowns makes it possible to discover unknown issues

early. Another way to cope with the unknowns is to minimize vulnerability.

6 Conclusion

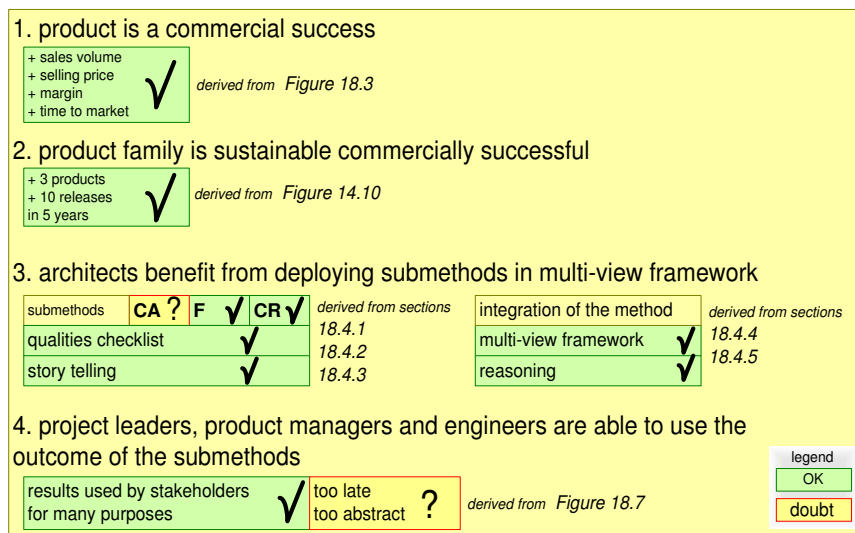


Figure 8: The conclusion of the case evaluation

The case has been used to test the hypothesis. The results of the evaluation are summarized in Figure 8. In many aspects the criteria are met:

Design and Integration According to Section ?? the quality of the design and the integration is a measure for Criterion 3, and the quality of the design is a prerequisite for Criterion 1. The quality of design was good (Section 2). The quality of the integration was also good (Subsection 4.5).

Criterion 1 The product is a commercial success (Section 3).

Criterion 2 The product family is sustainable commercially successful (Section 3).

Criterion 3 Architects benefit from deploying submethods in a multi-view framework (The submethods, the quality checklist and story telling were respectively evaluated in Subsections 4.1, 4.2 and 4.3). The integration of the method is supported by the multi-view framework and reasoning methods (the integration is evaluated in Subsection 4.5. The reasoning method, based on the threads of reasoning, is discussed in Subsection 4.4).

Criterion 4 The Product Creation team is used by many stakeholders for many purposes (Section 5).

Two evaluation aspects need more evidence or discussion. The submethods in the *Customer Objectives* and *Application* views are insufficiently covered by the case. In the next chapter other supporting evidence for the use of the submethods in these views will be supplied. The perception of some engineers that the results are *too late* or *too abstract* (Section 5) will be discussed further in Chapter ??.

References

- [1] Apple Computer, Inc. The Objective-C Programming Language. <http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/index.html>, 2003.
- [2] Gerrit Muller. Technology improvement plan. confidential internal report, June 1994.
- [3] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [4] Gerrit Muller. Case study: Medical imaging; from toolbox to product to platform. <http://www.gaudisite.nl/MedicalImagingPaper.pdf>, 2000.

History

Version: 0.9, date: April 7, 2004 changed by: Gerrit Muller

- Clarified the reasoning of the Vascular product group
- Clarified consolidation in CAFCR submethods
- added emerging behavior as source of integration problems

Version: 0.8, date: April 1, 2004 changed by: Gerrit Muller

- changed title into "Evaluation of the Architecting Method"

Version: 0.7, date: March 17, 2004 changed by: Gerrit Muller

- added reference to Objective-C
- reversed the order of the quality figures and descriptions
- changed status in finished

Version: 0.6, date: January 13, 2004 changed by: Gerrit Muller

- split Section Product and Design Evaluation in two sections.
- added source for customer oriented information.
- changed status in concept
- reformulated the solution of the *gap* problem
- some textual improvements

Version: 0.5, date: December 8, 2003 changed by: Gerrit Muller

- many textual improvements

Version: 0.4, date: December 4, 2003 changed by: Gerrit Muller

- many textual improvements
- changed status in draft

Version: 0.3, date: November 20, 2003 changed by: Gerrit Muller

- added granularity determination to the method coverage figure

Version: 0.2, date: October 1, 2003 changed by: Gerrit Muller

- adapted Figure conclusions
- added source and legenda to Figure product evaluation and to Figure design evaluation
- clarified the more objective assessment of the design

- added explanation about the political climate behind the technology improvement plan
- split the section "Product and design evaluation" in 2 subsections
- deleted paragraph about CR-bias

Version: 0.1, date: June 12, 2003 changed by: Gerrit Muller

- added introduction
- Added quality documentation
- added paragraph about credit for concept selection
- added usability
- added conclusion
- increased status to preliminary draft

Version: 0, date: June 11, 2003 changed by: Gerrit Muller

- Created, no changelog yet