14 The Role of Rationale in the Design of Product Line Architectures – A Case Study from Industry

J. Knodel, D. Muthig

Abstract: Product line engineering aims at an efficient production of variants mainly enabled by large-scale and systematic reuse of artifacts throughout all development phases. A product line's central artifact is its architecture that defines fundamental concepts, abstractions, and mechanisms that hold for all products of an organization (if successfully) for a long period of time. Therefore, key developers in organizations must fully agree on all decisions related to the definition of the product line architecture, as well as they must always reunderstand their rationales during architecture evolution. This chapter describes an industrial case of architecture evolution where one of the key mechanisms of an existing architecture was revisited as potential subject of change.

Keywords: architectural decisions; design; software architecture; product line architecture; product line engineering

14.1 Introduction

Nearly all organizations today develop and maintain more than a single product. Software organizations typically develop and maintain sets of similar products for different customers or market segments. This holds for organizations developing tailored systems individually for single customers, as well as for organizations developing products for mass markets. All products of an organization, however, are typically situated in the same application domain. Hence, these products share some common characteristics and thus can be viewed as a product line.

Product line engineering is the according development paradigm that differs significantly from traditional single system development. It aims at an efficient production of members of a product line mainly enabled by large-scale and systematic reuse of artifacts throughout all development phases (see [4] or [6] for definitions of software product line).

Product line engineering, thereby, analyzes the whole family of products rather than each product individually to systematically exploit commonality, proactively plan for variability and engineer variants efficiently. While performing these activities, the definition and design of the product line architecture is one of the most crucial activities. Since the architecture spans over the whole product line (including current products and envisioned or hypothetical products), the design decisions have a high strategic value and deep impact on the development organization. For this reason key decisions have to be well-founded and the rationales behind such decisions must be documented to be able to circumstantiate the decisions to the various stakeholders.

This chapter presents a real industrial case where an industry organization revisited one of the key mechanisms of an existing architecture while defining the product line architecture for its next generation of products. The evaluation of two alternative architectural concepts (i.e., a new candidate mechanism was identified to potentially replace a concept used in dozens of products of the previous generation), the common decision for one of the mechanisms, as well as the consistent documentation of the rationale was supported by an independent party, the Fraunhofer Institute for Experimental Software Engineering (IESE).

The approach followed during these activities is part of Fraunhofer PuLSE™ (Product Line Software and System Engineering, see Fig. 14.1)[15], which is presented in Sect. 14.2. Section 14.3 then describes the overall context by characterizing the analyzed TFT-panel product line – which is a main element in larger car radio and driver information systems product line – and, in particular, its Graphics component that is mainly impacted by the decision under consideration. Section 14.4 then presents details of the two alternative concepts, common design principles, and constraints on the decision. Finally, Sect. 14.5 concludes with experiences made and how the rationale approach additionally improved the quality of the product line architecture. Furthermore, impact and consequences of the decision are reflected with respect to both, concrete projects and the product line.

14.2 Approach

Fraunhofer's PuLSE method is a complete product line approach covering all life cycle phases and product line activities, as well as organizational issues or maturity models. Fig. 14.1 gives a complete overview of PuLSE components.

[15] PuLSE is a registered trademark of the Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany (see [2] and [5]).

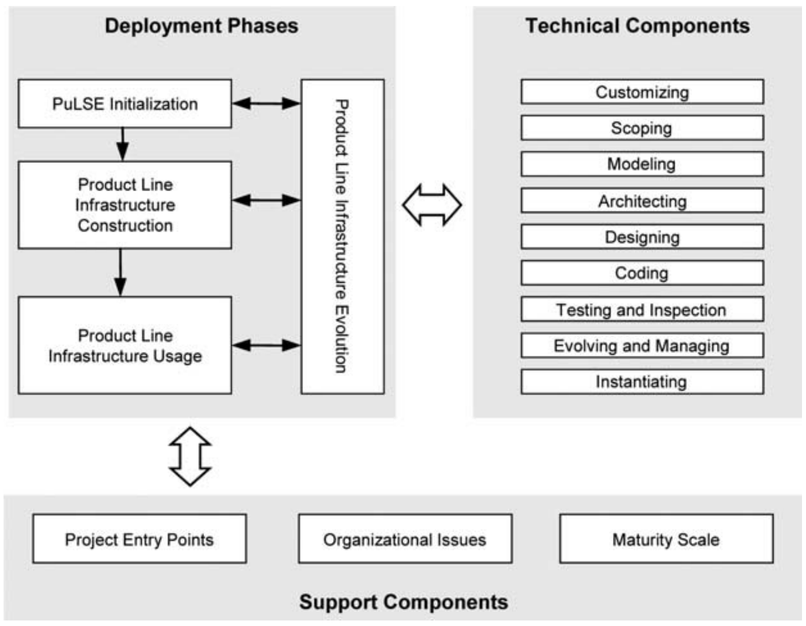

Fig. 14.1. PuLSE^{TM} Overview

PuLSE-DSSA ("architecting") is one of its technical components covering all activities and aspects related to product line architectures (see [1] for details). It is mainly concerned with the definition, evaluation, and evolution of product line architectures including the specification and execution of supporting reverse engineering activities. Overall, PuLSE-DSSA follows an incremental approach (Fig. 14.2 depicts PuLSE-DSSA).

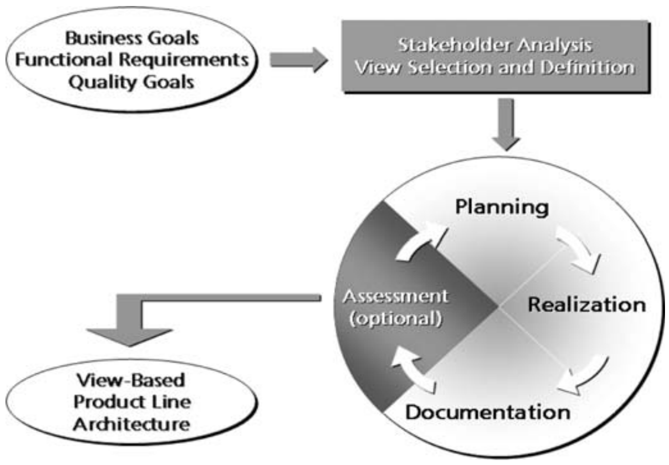

Fig. 14.2. PuLSE-DSSA

Technically, PuLSE-DSSA is a scenario-driven approach whereby "scenario" is defined consistently with the SEI's architecture assessment methods SAAM and ATAM (see [3] for a definition). That is, PuLSE-DSSA uses the same scenarios for defining architectures as it uses for assessing them.

While developing or evolving the architecture many decisions must be taken, which define or change the fundamental concepts, abstractions, and mechanisms that hold for all products of an organization (if successfully) for a long period of time. Therefore, key developers in organizations must fully agree on all decisions related to the definition of the product line architecture, as well as they must always reunderstand their rationales during architecture evolution. Consequently, an approach for managing architectural rationales is integrated into the general PuLSE-DSSA process, which is – consistently with the overall approach – scenario-based. Furthermore, it is prescriptive and intrusive because it is integrated with the decision process itself and thus improves decision makers and consequently also the final architecture.

Conceptually, the approach encompasses five steps, namely problem identification, criteria elicitation, evaluation and criteria assessment, decision making, and documentation:

– **Problem Identification**
  The first step is to identify, to understand and to learn about the problem that is linked to the decision. One the one hand one has to know the problem domain very well in order to be able to derive a sound decision, and on the other hand, the strategic goals that should be fostered by the decision have to be clear to everyone involved in the decision making process. These problem statements are often provided by the lead architect of the product line architecture. Usually they deal with crucial design problems in the development of the product line architecture. Other stakeholders having an interest in the decision have to be identified and their concerns have to be collected. Representatives for the different alternatives are required, so that each alternative has at least one advocate. The advocates have to allocate sufficient time for the next steps because based on their input, the decision will be drawn. The advocates are usually domain experts who promote or favor one of the solutions.
– **Criteria Elicitation**
  There are a lot of criteria related to design problems of product line architectures, and if there is enough time and effort, all these criteria have to be considered. In practice, however, there is always a certain project pressure, the work has to be done under tight time constraints, and the availability of experts and stakeholders is very limited. Therefore, it is

not efficient to assess all criteria. Some criteria have a higher priority than others, some cannot be influenced by the project management, developers or engineers, and again others are equivalent for all alternatives. For this reason, this step has the goal to elicit the important and determining criteria, which are based on the high-priority scenarios of PuLSE-DSSA. Usually those criteria divide the alternatives from each other. By eliciting these criteria, the effort for decision making is reduced, but nevertheless focused on the aspects that swing the decision. For each identified relevant criteria, one assessment table is created (see Table 14.1 for the template). Preceding the table, an explanation of the criteria is given, so that the meaning is recorded. The fill in of the table with content is done in the next two steps.

– **Evaluation and Criteria Assessment**
  Each advocate presents his favored solution to a moderator (the product line architect should participate here, but it is not mandatory) and explicitly addresses the criteria with pros and cons. The moderator records the arguments and the reasons why something is an advantage or a drawback. Furthermore, the moderator should actively participate by asking clarification questions. The same is done for other alternatives one after another. This results in filled criteria assessment tables, with the exception of the result cells.
– **Decision Making**
  The decision making step is conducted in a workshop where the product line architect and the advocates together discuss the arguments for the different alternatives. The goal of this joint workshop is to agree on the best alternatives with respect to the assessed criteria. The criteria are discussed stepwise so that obscurities are smoothed out and opposite views come to an understanding of all stakeholders. The result documents for each criterion, which of the alternatives is considered as best and for which reasons. In the end, the stakeholders agree on the best candidate based on the individual criteria. Furthermore, potential action items for future improvements for the selected alternatives can be identified by comparing it to other alternatives. The moderator summarizes the final decision and action items. Ideally, all involved stakeholders agree upon the decision.
– **Documentation**
  The documentation of the final decision and the aspects that led to it is an ongoing activity that accompanies all of the above steps. The final report contains the problem (i.e., why there was need to decide something), the alternatives (i.e., which solutions where considered for the decisions), the criteria (i.e., what was assessed, and what were the main

drivers leading to the final decision), the arguments (i.e., what were the pros and cons of each alternative), and of course, the final decision, based on the important criteria (i.e., the selected solution), and derived action items (i.e., what has to be taken care of in future). The documentation can be used to inform other people of the organizational unit, and to put the decision on a firm footing, in case the higher management or other people challenge the decision made.

Table 14.1. Template for criteria assessment

	Alternative 1	Alternative 2
PRO	Alternative 1 – Pro Argument 1	Alternative 2 – Pro Argument 1
	Alternative 1 – Pro Argument 2	Alternative 2 – Pro Argument 2
CON	Alternative 1 – Con Argument 1	Alternative 2 – Con Argument 1
	Alternative 1 – Con Argument 2	Alternative 2 – Con Argument 2
→	Result and the derived decision based on the arguments	

The approach as described here usually has to be adapted slightly to the organization and the available resource constraints. Depending on the architectural design problem, the number of alternatives and elicited criteria can vary. The main goal is to consider only adequate alternatives (not everything is feasible or appropriate) and to select only criteria that are central to the decision (not everything is important or relevant).

14.3 The TFT-Panel Product Line

The case study was conducted in a larger context with the goal of the migration of an organization towards product line engineering. The activities were driven by the Fraunhofer Institute for Experimental Software Engineering (IESE) in cooperation with one of its industry partners. Product line concepts were introduced incrementally in pilot projects whereas one of the projects was concerned with the construction of the panels for 1-DIN navigation systems. 1-DIN navigation systems cover pure car radio products and navigation system products that provide route guidance solutions. Such a system supports different data mediums (e.g., CD, DVD, HDD containing music data, voice data, video data, or navigation data) that constraint the features offered to the driver.

Fig. 14.3 depicts a sketch of the major components of a 1-DIN navigation system, which fits into the standard car radio slot of most vehicles. It is distinguished (physically and logically) between two embedded systems, the silver box and the panel, but both are closely related to each other. The silver box contains the CD or DVD drive, the navigation processor, and all other parts not related to the display. The panel has an integrated display for visualizing radio and navigation functionality. It consists of the display, different buttons and a physical interface to the silver box. Engineering panels deals not only with software, but also with aspects of hardware and mechanics.

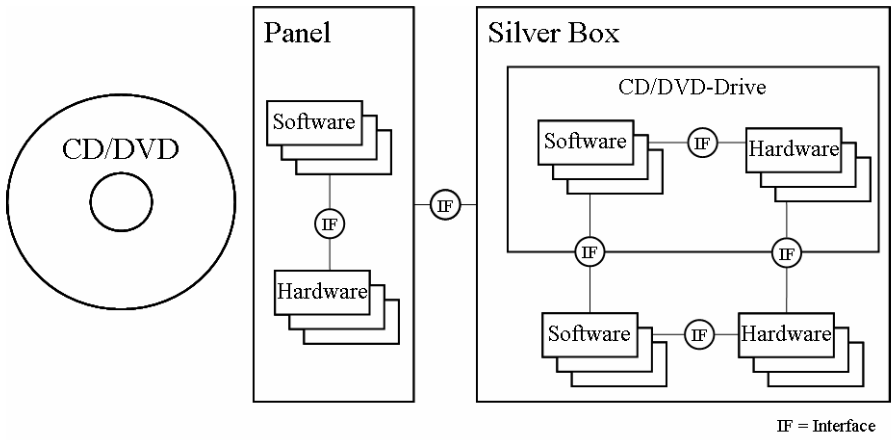

Fig. 14.3. Panel of the *1-DIN navigation system*

Most previous products of the company used monochrome panels only. For these monochrome panels, the complete information to visualize was provided by the silver box's processor. Such panels can be regarded as pure display facilities. On the contrary, the current development addresses a new type of panel, TFT-panels that become part of 1-DIN navigation systems. Such TFT-panels obtain some "intelligence" by having more computing power inside the panel. The larger context of the case study was to define a product line of TFT-panels that serves the current development and all upcoming car radio and navigation systems of the next years.

When developing the product line architecture for the TFT-panel product line, we reviewed the key concepts, mechanisms, and components of the existing monochrome panels, in order to decide, where reuse is possible and to identify the need of adaptation, and the need for the development of new components.

14.3.1 The Role of the Graphics Component

The Graphics component is one of the key components of the TFT-panel architecture. It is responsible for the communication between the silver box and the panel as well as for the composition and visualization of the correct output to the driver in the car. The interaction with the driver takes place via predefined masks. A mask is defined as a collection of graphical elements and positioning information (e.g., text fields, bitmaps, buttons, lists, labels). The graphical elements contributing to a mask are divided into two groups:

– *Static information relevant only for the Graphics component*. These elements are static and will not change once the system is in operation. It is decided at design time what these elements are and how they will look like. The elements are only known to the Graphics component. The static elements include placeholders for dynamic information coming from the silver box. Examples include fonts, fix bitmaps, fix text (in all supported languages), background pictures, etc. This graphical information will not change once the TFT-Panel is in service.
– *Dynamic sequence control information*. The elements related to the sequence control are dynamic information that is context dependent. The dynamic elements are computed by the processor in the silver box based on the current context (e.g., navigation map cut-outs are dependent on the position of the car). They complete the static masks held in the Graphics component with context relevant information that is only known at run time. Examples for such elements are headlines, radio station names, frequencies, town names, and street names.

14.3.2 Management and Transfer of Graphical Data

The main architectural driver for the Graphics component is the minimization of the data flow between panel and silver box. The communication interface is an SPI (Serial Peripheral Interface) protocol offering logical channels and encapsulating the hardware connection between the panel and the silver box.

Since the SPI bandwidth is limited, the amount of data transferred and the number of messages sent has to be kept as low as possible. This performance criterion was already the main architectural driver for the development of the Graphics component for the monochrome panels of previous products. Fig. 14.4 shows an example how the human—machine interaction (HMI is running in the silver box) communicates with the Graphics component (running in the panel) via the SPI interface. The messages received in the panel are parsed and decoded by the Graphics component, which then invokes basic painting functionality provided by a low-level Graphics API.

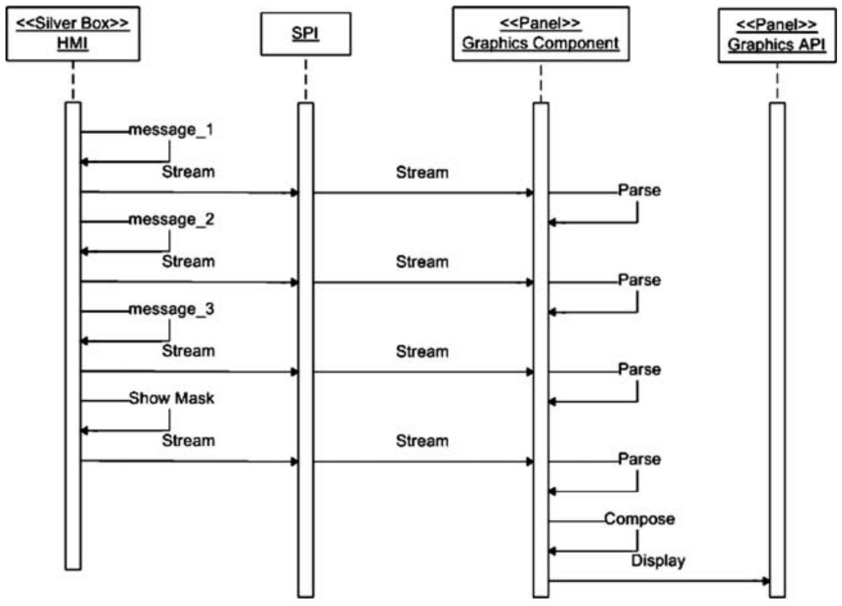

Fig. 14.4. Communication between Graphics component and Silver Box

When evaluating the existing architecture of monochrome panels for its reuse potential, the question arose whether the mechanisms applied for the Graphics components of monochrome panels (called Mask-Oriented Communication, MOC) can serve as well as a basis for the TFT-panel product line architecture. The suitability was questionable because of the required genericity of the Graphics component (i.e., to serve all variants of the product line), the uncertainty (i.e., to provide sufficient flexibility for anticipated changes), the changed technology (i.e., new type of panel, TFT instead of monochrome), and the increased computing power that enabled new features (i.e., a processor inside the panel to take over graphical drawing functionality). Some engineers of the company came up with an alternative mechanism for the Graphics component (called Framework-oriented Communication, FOC). Since this new mechanism was developed from scratch, no real project experiences were available, and the architects were doubtful about the reached maturity of the FOC mechanisms because only a first prototypical implementation existed at that moment. The project management became aware of the competing alternatives and demanded a justified decision, especially addressing the technical problems and product line aspects.

Both mechanisms followed the general communication principle of using a logical channel of the SPI protocol to connect the panel to the silver box. The data model of both mechanisms was practically the same, that is, identical sets of graphical elements were supported by both alternatives. The general mode of operation is that the silver box sends messages with parameters that contain the information to be displayed and the Graphics component then parses the messages and displays the right graphical elements. The next sections will introduce the two alternatives (MOC and FOC) in more detail. The mechanisms mainly differ in what messages are transferred (e.g., the content, format, the size, and the number of the messages), how the interpretation of these messages is addressed, and how the different masks visible to the driver are specified, constructed, managed, and maintained.

14.3.3 Alternative 1: Mask-Oriented Communication (MOC)

The first alternative, which was used for previous projects constructing monochrome panels, implements a mask-oriented transmission mechanism. The graphical elements are transferred one after another and then put together in the Graphics component of the panel.

Each message (e.g., ID_Tuner, ID_Frequenz "ffn") contributing to a mask is streamed directly over the SPI interface. The Graphics component then parses the messages and finally decodes the received information (i.e., selection of masks, references to bitmaps, replacement of dynamic information). Masks are composed in the panel's background until the message "ShowMask" is received by the Graphics component. The Graphics component then initiates displaying the graphical information by using services of the low-level Graphics API, which then triggers the display itself.

The masks are built with the help of a construction kit that consist of a row of graphical elements. Mask specifications are captured in header files that are shared between the HMI running in the silver box and the Graphics component running in the panel. Thus, the HMI can directly address all graphical elements of the masks. Data required by different products is configured via a tool that ensures consistency of data. The tool generates the header files, which typically have to be adjusted few by developers, and takes over consistency checks to ensure the match of the header files in the silver box and in the panel.

The MOC mechanism is realized in C and has been applied to several projects. The developers are experienced in applying this mechanism and know about the weaknesses and its pitfalls.

14.3.4 Alternative 2: Framework-Oriented Communication (FOC)

The FOC alternative follows the principle of object-oriented GUI frameworks as they are often implemented for desktop PCs. It is based on a model, view, controller pattern whereas the view resides on the panel and the controller on the silver box. The framework offers certain standard ways of constructing, modifying, manipulating, and managing graphical objects. The communication between silver box and panel takes places over abstract interfaces that are implemented by each graphical object.

In the FOC mechanism, a mask is decomposed into a hierarchy of structural elements. Terminals of the hierarchy are basic elements such as buttons, list boxes, text fields, text label used to represent concrete GUI objects, which are then combined with structural elements (e.g., columns, rows, widgets, buttons) to compose complex masks. Each of these objects is responsible for its own representation (a paint method invokes the respective functionality). Some masks require special extensions; these specialties are realized by inheritance from the most similar element. All graphical objects inherit from an abstract class and are positioned into a hierarchy relative to their direct parent.

Messages sent from the silver box to the panel are limited to the interfaces of the abstract classes. The parameter string contains the required information to display a mask, or to change the status of displayed graphical objects (e.g., scrolling, marking of an element). The realization of functionality is realized by the base elements (e.g., activation of a button within a button widget) from which other elements can inherit. The Graphics component takes over the serialization of graphical objects in order to ensure a smooth data exchange with the processor in the silver box.

Currently, there is no tool support for the construction of the masks, and no practical, real project experiences were available. The implementation of the FOC mechanism prototype was realized in the C++ programming language, and it adopted a lot of object-oriented principles.

14.4 Concept Assessment and Decision Making

We applied the concept assessment and decision making steps as described earlier to derive the decision whether to adapt the Mask-Oriented Communication or to extend the prototype of the FOC. IESE held the role of the moderator, while the advocates and the product line architect were part of the industry partner's development organization. The two alternative mechanisms were competing, each favored by different groups of developers. The concern of the product line architect was the question whether to reuse the “older” mask-oriented mechanism or to realize the “new” framework-oriented mechanism.

The impact of the decision to be made was regarded as critical by all stakeholders (including developers, architects, and project management) since the Graphics components of the TFT-panel product line is supposed to serve all upcoming variants of car radio and navigation system products in the future. The decision and the rationale leading to it have to be well understood and the key stakeholders have to fully agree on the decision made to pull together towards the success of the envisioned product line and its underlying architecture.

14.4.1 Assessment

In the criteria elicitation step, it became clear that the assessment criteria were threefold: technical constraints, general aspects, and product line related criteria. General aspects like available resources or time limitations were for both mechanisms the same, as well as some technical aspects like energy resource consumption or size of memory required on the TFT-panel. We left out these criteria and selected only relevant technical (e.g., SPI messages, tool support) and product line related aspects (e.g., support of new technologies, specialties and configurability, map processing, integration with car radio products).

One of the main criteria in favor for the FOC mechanism was the fitness towards very probable technology changes; the most likely are the introduction of touch screen functionality replacing mechanical buttons by on-screen soft buttons and the replacement of the SPI by another interface that enables faster communication between silver box and panel. The TFT-panel product line architecture has to be flexible toward these changes in underlying technologies. For instance, for the touch screen functionality, the Graphics component has to provide certain mechanisms to enable an input channel from the panel and to interpret the touch screen buttons pressed. Table 14.2 shows the arguments and the result of the assessment for the criterion “Fitness for New Technologies”.

One of the main criteria for the MOC mechanism was the “Tool Support” criterion, as the excerpt in Table 14.3 shows.

Table 14.2. Assessment: Fitness for New Technologies

	Mask-Oriented Communication (MOC)	Framework-Oriented Communication (FOC)
PRO	replacement of the communication carrier: no rework is expected since transmissions are operating on a logical SPI channel	replacement of the communication carrier: no rework is expected since transmissions are operating on a logical SPI channel
		touch screen: direct mapping of buttons to graphical elements facilitates handling and resolution of touch screen events
	touch screen buttons: fixed button positions can be handled with a work-around	touch screen buttons: Position Change of buttons will require no rework in the silver box.
CON	Touch screen: No support for touch screens events	touch screen: realization is not yet completely realized for this mechanism
	touch screen buttons: major rework in the silver box and the panel	touch screen buttons: some rework for in the panel
→	The FOC is the more future-proof mechanism since it already prepares the support of touch screen functionality.	

14.4.2 Results and Decision

We continued to evaluate both mechanisms in the same way for all criteria. The arguments (pro and con) for each criterion were recorded one after another, and the results were documented together with the rationales. Then all criteria were aggregated and put together to get the whole picture.

The result shows that the framework-oriented communication mechanism was rated overall as the better alternative in total. Furthermore, the FOC mechanism had higher ratings for the criteria with the highest priority. Those criteria were concerning product line related aspects (e.g., Fitness for New Technologies, Cross Organizational Product Line, Specialties and Configurability, Development Process Integration). Figure 14.5 depicts an overview on the assessed evaluation criteria. Overall, the FOC mechanism was evaluated to be more future-proof in this point than the already existing MOC. A side-effect was that all participants of the final workshop were able to understand each mechanism in detail, including advantages and drawbacks.

Table 14.3. Assessment: Tool support

	Mask-Oriented Communication (MOC)	Framework-Oriented Communication (FOC)
PRO	target group: HMI designers.	target group: developers.
	tool chain: Existing tools support simulation, header generation for masks, consistency checks	tool chain: existing tools support the simulation of panels
	experiences: applied in previous projects.	
CON	tool state: development freeze in 1997, no experts are available, proprietary data format.	tool state: no tool support yet, high risk of new tool development (unclear impact on projects and organization).
	tool evolution: complex and costly tooling.	
→	although the tool support for the MOC mechanism has some weaknesses, it contributes significantly to the product development. Code generation and consistency checks support the management of a large number of product line variants.	

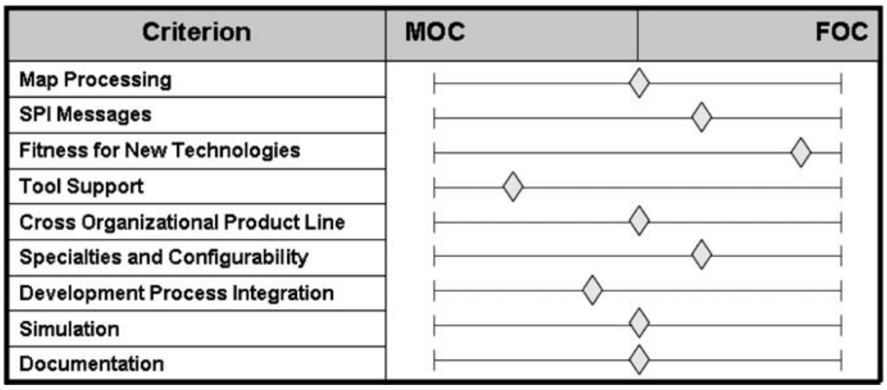

Fig. 14.5. Assessment results

Finally, all participants of the workshop (including both advocates and the product line architect) agreed on the framework-oriented communication mechanism as the solution to be implemented for the Graphics component of 1-DIN navigation systems with a TFT-panel. As shown in the for "Fitness for New Technologies" criterion, the advantages of the FOC mechanism outplay the MOC mechanism. In addition to the decision, some results of the workshop were identified as action items and improvements to the mechanism inspired by experiences with the other alternative.

The learning effects were gained by the common understanding of both mechanisms in detail. In particular, review dates were arranged to monitor the realization and implementation of the new mechanism. Tool support to comprise mask generation and consistency checks was identified as an important issue, and therefore development of tooling was scheduled. In addition to these product specific prerequisites, further architectural design goals for the TFT-panel product line have been documented. Both mechanisms were not yet adequate to address these architectural requirements. For instance, map processing features that generate the map within the panel and no longer in the silver box were not yet prepared, and the alignment of the overall development processes were an open issue with some impact on the TFT-panel product line. The rationales leading to the resulting Graphics component (and the reason for rejecting the other alternative) have been recorded. In case the decision is challenged by others developers or the higher management or other stakeholders not involved in the technical decision process, it can be defended based on the documentation. To retrace and understand the rationales for, and to be able to relate to the decision made, the criteria, the arguments and eventually the decision are documented. This enables as well a good start for new developers when learning about the product line architecture and its underlying concepts.

14.5 Conclusion

In the design of product line architecture the architects will face hard decision, where they are not immediately able to say one solution is more appropriate than the other. These decisions have a high impact and the consequences have to be covered not only by a single product, but the whole product line. When facing such a problem, it pays off to invest some effort and time to derive a sound, well-founded decision and to record the rationale and the reasons on which the decision is based on. Another benefit is the better understanding and the improvement of the quality of the selected solution gained in the workshop, where the alternatives are discussed in detail.

The approach we introduced here has not the goal to discuss all facets of the alternatives, but to focus only the key aspects that are of a high importance to the strategic goals of the product line and have a high architectural relevance. By reducing the evaluation criteria in this way, only limited effort is needed and the expert involvement can be reduced to a reasonable size.

One aspect in the introduction of product line engineering in the pilot project was to decide about the mechanism for the Graphics component. The two candidate mechanisms (MOC and FOC) were compared and assessed with respect to well-defined criteria. In examples, we highlighted the strengths and weaknesses of the two mechanisms, and the reasons for the decision were explained. The main reason for selecting the winning mechanism was its superior characteristic with respect to anticipated requirements of the envisioned product line. The common agreement (even the advocate of the other mechanism agreed on it) is a major gain for the next steps in the realization of the product line. A positive attitude toward the new mechanism was established and the stakeholders were highly motivated to start with the implementation. The explicit documentation of both mechanisms helped us to identify issues that must be addressed in the near future and topics that have to be addressed by the product line in a mid-term time frame.

The importance of rationale in product line architecture is critical since the product line architecture is the foundation for all derived variants. Next to putting the decision on a firm footing, the whole development team (and new members) can learn on the one hand about the design problem faced and the potential alternatives, and on the other hand about the reasons that lead to the decision made. By documenting it, the decision serves as well as justification in case the higher management or other architects challenge the decision at a later point in time. The approach helped to select the FOC mechanism and the experiences with this mechanism so far are very promising.

References

[1] Bayer J, Forster T, Ganesan D, Girard J-F, John I, Knodel J, Kolb R, Muthig D (2004) Definition of Reference Architectures based on Existing Systems, Fraunhofer IESE Technical Report (IESE-Report 034.04/E), Kaiserslautern

[2] Bayer J, Flege O, Knauber P, Laqua R, Muthig D, Schmid K, Widen T, DeBaud J-M (1999) PuLSE: A methodology to develop software product lines, in: Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR'99), ACM, Los Angeles, CA, USA, May, pp. 122–131

[3] Clements PC, Northrop L (2001) Software Product Lines: Practices and Patterns, SEI Series in Software Engineering, Addison-Wesley, Reading, MA, August

[4] Clements P, Kazman R, Klein M (2002) Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, Reading, MA

[5] PuLSE (2005) Product Line Software Engineering http://www.iese.fraunhofer.de/PuLSE/, Fraunhofer IESE, Kaiserslautern

[6] Weiss DM, Lai, CTR (1999) Software Product-Line Engineering: A Family-Based Software Development Process, Addison-Wesley, Reading, MA<AQ: Refs. [2,5] are not cited in text>