Variation Mechanisms and Multi-view Architecting in Platform-based Product Family Development

Wijnstra, Jan Gerben

PART III PRODUCT FAMILY DEVELOPMENT AND EVOLUTION

Chapter 7
Critical Factors for a Successful Platform-based Product Family Approach

Abstract

The notion of software product families is becoming more and more popular, both in research and in industry. In presentations of ideal applications, the benefits of product families are stressed and very little attention is paid to possible problems. However, in practice, it is important to know what such problems could be and how they could be dealt with. In this paper we want to identify some of the most critical issues, and explain how they can be handled or even avoided. This will be done on the basis of experiences gained with three industrial product families for professional markets. The focus will be on product families that use platforms with reusable assets for the development of the family members.

7.1 Introduction

Products in the various embedded system markets are becoming more complex and more diverse, and must be easily extendible with new features. Important factors are a short time-to-market, low development costs and high demands on the quality of the software. One way of meeting such requirements for a range of products is by defining a product family with a shared architecture and reusable assets organized in a platform.

In practice it is relatively difficult to introduce a product family approach. Many initiatives do not achieve their goals or even fail because their impact and potential problems were not properly identified. Introducing a product family approach is not a matter of simply using a new technique, i.e. a platform with reusable components. A product family approach involves more than architectural issues alone; it also affects a company's business, processes and organization. For example, the product-management, marketing and project-management departments must be familiar with the approach, requirements management must take account of it, etc. If no due allowance is made for all these aspects, then the chances of the introduction of such an approach proving a success will be small.

In this paper we discuss some the issues that are of importance in the context of product families on the basis of three industrial cases. To structure our discussion, we will use two dimensions, as shown in Figure 7-1.

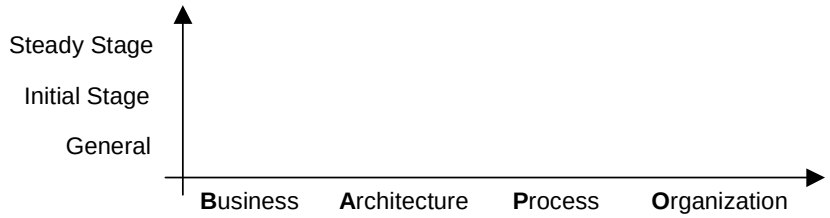

Figure 7-1 – Two Dimensions

The first dimension comprises Business, Architecture, Process, and Organization (BAPO), which is a reasoning framework introduced in [2] and [70]. It places architecture in the context of business, process and organization; see Figure 7-2. It also defines logical relations between them (the arrows in Figure 7-2). For example, processes are ideally defined in such a way that they optimally support the realization of the architecture. The architecture is further refined into five views (CAFCR, see [70]):

- **C**ustomer (the customer's world)
- **A**pplication (applications important to the customer)
- **F**unctional (requirements for a system used in a customer application)
- **C**onceptual (architectural concepts of the system)
- **R**ealization (realization technologies for building the system)

The second dimension comprises issues that apply to a product family approach in general, issues in the initial stage (when a product family approach is introduced) and issues in the steady stage (when it has already been applied for several years).

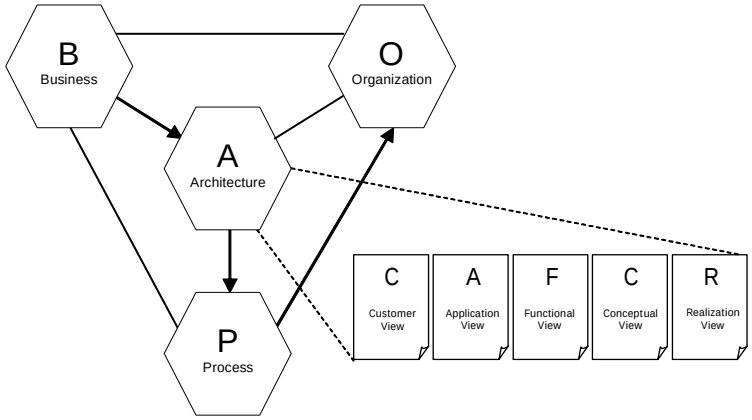

Figure 7-2 – BAPO/CAFCR Framework

The paper is organized as follows. In section 7.2, the three industrial cases will be briefly introduced. Sections 7.3 through 7.6 will cover the BAPO issues. Each of these sections will discuss general product family issues and will conclude with a subsection on initial stage issues (in this paper, no special attention will be paid to the steady stage). Conclusions can be found in section 7.7.

7.2 Three Cases

In this paper we will use experiences gained with three industrial product families in two domains, viz.:

- Telecommunication Switching System (TSS)
- Medical Modality System (MMS)
- Medical Imaging System (MIS)

The TSS product family covers telecommunication infrastructure systems. This family has been designed for niche markets with a high variety of features. A key requirement was to achieve a reasonable price even with a low sales volume. Products of this family include public telephony exchanges, GSM radio base stations, technical operator service stations, operator-assisted directory services and combinations of the aforementioned.

The other two product families come from the medical domain. Philips Medical Systems builds products that are used to make images of the internal parts of the human body for medical diagnostic purposes. The different types of equipment used for acquiring images are called ‘modalities’, for example Magnetic

Resonance (MR), X-ray, Computed Tomography (CT), and Ultrasound (US). In this paper we shall take a product family for one such a modality as a case, called MMS. There are also similarities across these modalities. In order to benefit from these similarities and increase the synergy a platform is being made that is used by several modalities (including the MMS family). This platform focuses on medical imaging functionality, and is called MIS.

Although these families have different scopes and cover different domains, they share a number of important characteristics:

- complex, software-intensive products;
- sold in small numbers, but having a lot of diversity;
- professional products;
- long lifetime and support of products (10-15 years);
- software updates in the field;
- extensible with new features;
- a customer typically has a number of products from one family;
- standards for interconnection;
- relatively mature and stable domains;
- previous experience with such products.

It must be noted that the three cases described here are product families intended for professional markets. These markets and the products for these markets are somewhat different than for example a high volume market with customer appliances. Even so, we are of the opinion that the BAPO issues discussed in the next sections are also relevant for product family approaches in other domains.

7.3 Business Aspects

Business concerns a company's mission and market. A strategy must be defined within the business that supports the mission. Several elements make up a company's strategy. In this paper we focus on a product family platform as such a strategic means. A product family platform should thus be seen as a means and not as a goal.

7.3.1 Why Platform-based Development?

According to the BAPO framework, the rationale for a platform architecture and platform-based development can be found in the business. So, what are the business drivers for applying such an approach? These drivers can be external, i.e. matching a customer's need, or internal, i.e. supporting the development of the products internal in the company. The main reasons mentioned in the context of the three cases include:

- **Variation on a shared basis.** The systems are being made in small numbers. The customers on the other hand have their own specific demands. This leads to a range of similar, yet different systems.
- **Feature propagation.** A single customer will usually own not one telecommunication switching or medical system, but several. As a result, a customer may request that a feature that is present in one system be provided in other systems, too.
- **Timely availability.** In order to be competitive, new features must be timely introduced into the market. Deadlines agreed upon with the customer must moreover be met.
- **No unnecessary diversity.** No unnecessary diversity must be exposed to the systems' users, who will usually have to deal with several similar systems. For example, a physician may have to be able to operate different types of X-ray systems. And a field-service engineer will have to know how to service various systems in the hospital. If the systems are similar, less training and fewer manuals will be needed to use or service the systems, which will lead to greater efficiency.
- **Limited development costs, shortage of skilled labor.** In the current economic situation, it is important to be able to limit development costs so that products can be offered at reasonable prices. There may also be a shortage of skilled staff needed to develop the systems.
- **Software complexity (>100 man years, $>10^6$ lines of code).** The systems we are talking about are complex and contain over 1 million lines of code. Such systems cannot be developed from scratch each time.
- **Higher quality.** The functionality within the platform will be reused in difference places, which will increase confidence in it. In addition, the functionality's testability will increase with regression tests, etc. This however holds only for unmodified reuse; if components need to be altered, this will apply to a much lesser extent.

- **Interoperability.** The individual systems within a family must be able to interoperate with each other, because a single customer will usually use several family members. This can be ensured by including interoperability functionality in the platform itself.
- **Evolvability.** Because of the large investments, the same basis must be used for building the products over relatively long periods of time. Because of new requirements and new technologies, this basis must evolve over time. If this basis consists of a component-based platform, the individual components will be able to evolve gracefully, assuming an appropriate separation of functionality is chosen.

7.3.2 Product Lines and Product Families

In this paper we use the term product family as a set of related products that are realized on the basis of a shared architecture, using assets from a platform. This focuses on the internal part of the business. For the external view, we will use the term product line, which represents a set of related products for the business' customers. It is often assumed that a product line is related to a product family on a one-to-one basis. For example, the definition of product line in [14] mentions both a market segment and a common set of core assets (platform). There are however more possible relations between product line and product family, as illustrated in Figure 7-3.

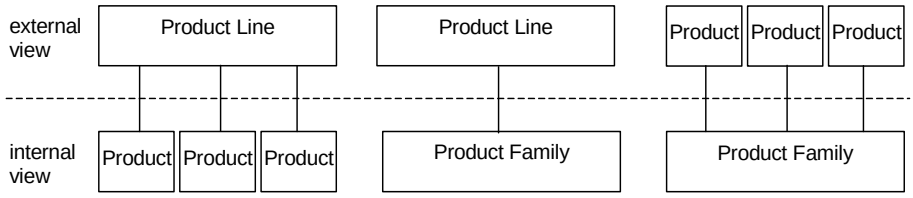

Figure 7-3 – Product Line and Product Family

The middle picture represents the 'common' situation. Another possibility is that a set of products is commercially marketed as a product line, while the individual products are developed independently (left picture). The other way round, a set of products may be developed as part of a product family, while the products are marketed individually (right picture). We will return to this later.

7.3.3 Value and Levels of Reuse

In [49], four levels of reuse are described; see Figure 7-4. These levels range from ad-hoc reuse, which means that a developer reuses code when (s)he sees

suitable code, to strategy-driven reuse, with which the reuse is aimed directly at supporting the company's strategy. This means that business people will also be involved when taking the decision to pursue strategy-driven reuse. Strategy-driven reuse means for example that by having a reusable platform, it will be easier to enter new adjacent markets.

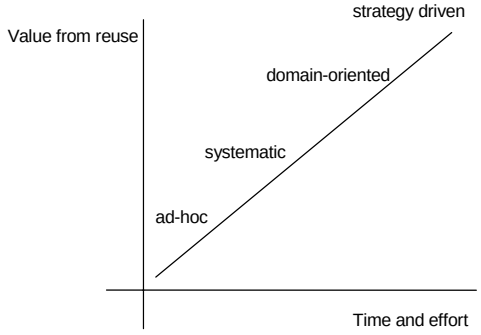

Figure 7-4 – Levels of Reuse

As graphically shown in Figure 7-4, the time and effort needed to achieve strategy-driven reuse are greatest because of the large scope and impact of this form of reuse. The option of applying such reuse must therefore be carefully considered. If, say, you have not yet worked out your strategy in full detail, it would be dangerous to try out such a new way of working, not knowing whether it will prove to be effective.

The three cases considered in this paper all have a flavor of strategy-driven reuse. The MIS platform, for example, has to support important business goals, it must allow for an integrated product portfolio and a common look-and-feel across products. The TSS platform was developed to be able to quickly enter new niche markets with low-priced products.

When starting to apply a product family platform, it is important to also embed it in the company in order to obtain optimum results. The real value of applying a platform approach in a certain context is hard to predict. There are still no clear rules to exactly predict the value of this kind of reuse. In [106], some information is given on the costs and benefits of a platform approach. Since a product family's scope must be known in order to determine an approach' value, we will now discuss scoping.

7.3.4 Scoping of Product Families and Platforms

In this paper we make a distinction between family architecture and platform architecture. By platform architecture we mean the architecture that is used to

build the reusable assets within a platform. By family architecture we mean the shared architecture with which the family members must comply. The rules of the family architecture must enable reuse of platform components, but may also contribute, for example, to more commonality in the family so that it will be possible to merge individual systems in the future. The family architecture focuses on shared architectural issues of the family members, the platform architecture focuses on the platform's internal structure.

The product line scope must be determined on the basis of (external) market reasons. The family and platform scopes depend on external and internal issues. Some of the reasons mentioned for a product family approach, e.g. the feature propagation (external driver), can be serviced well if there is a one-to-one relation between product line and product family. However, if the focus is more on cost reduction (internal driver), and the family platform focuses on reuse in technical or infrastructure subdomains[11] and not on application subdomains, this product family may show no external similarities, and thus not be considered a product line as illustrated in the right picture in Figure 7-3. So, internal business considerations can also lead to a product family approach. All these consideration must be taken into account when roadmapping a platform within a business strategy. The TSS and MMS product families are also marketed as product lines. The MIS case lies somewhere between the middle and right pictures of Figure 7-3; the products are part of the same portfolio, but the reuse currently focuses on technical and infrastructure functionality. In the future, the platform focus will evolve more towards application (external) functionality.

When a family scope has been found, it must be iteratively further investigated to find out how it can be supported by a platform. The main shared functional parts must be identified on the basis of the family scope. The main architectural rules and concepts supporting the qualities of the products within the family must also be considered. The platform's scope must be such that the architecture-shaping qualities are shared for this scope. If the platform scope includes too diverse systems, a platform approach may become counter-productive. An initial architecture can be generated on the basis of the architecture-shaping qualities and be used for a first cost-benefit analysis. These considerations lead to the definition of a platform scope within the product family's scope.

---

[11] By 'subdomain' we here mean a subarea within a system which requires some specific knowledge, for example the image processing subdomain, or the image acquisition subdomain. These subdomains may relate to application knowledge relating to the user, to technical knowledge relating to the peripheral hardware or computing infrastructure knowledge.

Another, more technical, choice is the asset content of the platform; are all assets been included (including product-specific ones), or only the assets that are used within several family members. Figure 7-5 graphically represents the various platform and product scopes. The family functionality is represented as white rectangles divided into a number of subdomains, the platform functionality is represented as gray rectangles.

- The TSS platform contains all the components of all the systems in the family; the family architecture is the same as the platform architecture. The right components must be selected for a specific product.
- The MMS platform contains components relevant to two or more systems in the family; only family-member-specific functionality has to be provided for each family member. The family architecture is more or less the same as the platform architecture.
- The MIS platform contains components for the imaging subdomain that are relevant to two or more systems in the family; each family member still has to add product-specific functionality for this subdomain and all functionality for the other subdomains. The family scope is broader than the platform scope.

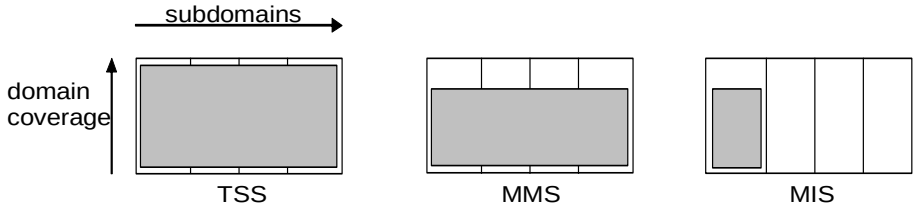

Figure 7-5 – Platform Coverage of Product Functionality

On the basis of the two dimensions used in Figure 7-5, the three cases can also be classified in a table, as shown in Table 7-1. The columns indicate whether the platform covers all the subdomains of the products in the family or only a subset. The rows indicate whether the platform contains all the products' functionality, or only the generic functionality (i.e. whether it applies to at least several family members).

	subset of domain	**complete domain**
generic functionality	MIS platform	MMS platform
all functionality		TSS platform

Table 7-1 – Platform Types

So far, we have assumed that a product family is based on one platform. But in view of the large scopes of product families with diverse family members, it is also possible to build a product based on several platforms, as shown in Figure 7-6. On the left-hand side, a platform is used to make another platform, which is again used to make the final products. This situation holds for the combination of MIS and MMS; the MMS platform is built using components from the MIS platform. The users of the MMS platform use it as a single platform and do not see the MIS platform as a separate one. The right-hand side of the figure illustrates a situation in which products are built on the basis of two platforms. Such platforms will usually focus on different subdomains of the family.

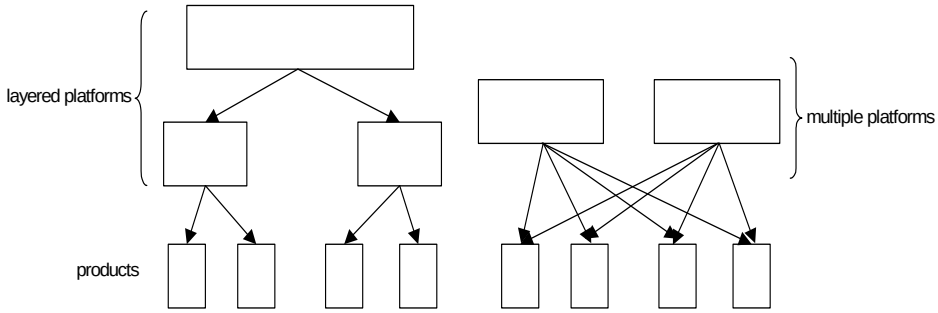

Figure 7-6 – Layered and Multiple Platforms

Depending on an organization's size and market scope, one or more platforms can be applied to support its mission. When only one platform is used for a large part of the business, there will be a risk of the platform capabilities being overstretched. We experienced this in some respect in the case of the MMS platform, which covers high- to low-end products, leading to a broad range of architectural requirements. This is hard to realize within a single platform. When several platforms are used, it must be made clear how the scopes of the various platforms complement each other.

7.3.5 How Stable is a Platform?

It is important to note that a platform should not be regarded as a stable unchangeable base. Changes will usually have to be made in a platform to meet the requirements of a new product. There should be no risk of missing market opportunities for the reason that 'they could not be incorporated in the platform'.

7.3.6 Initial Stage

Some business-related issues that will have to be considered in the initial stage already are:

- **Maturity and stability of the market.** It is important to note that all three cases are based on existing systems intended for relatively mature and relatively stable markets. So the important domain concepts are known and are not likely to change very fast. This is very important when a platform approach is introduced, since such an approach forms a (reusable) consolidation of these concepts, and should not innovate too much. However, a platform should enable the business to enter adjacent markets if necessary.
- **Vision of the future.** A business roadmap is needed to describe what is expected from the platform in the years to come and how it fits in the plan for the release of products. The future expectations for the platform help to identify evolution requirements, required staffing, how it should be introduced, etc.

7.4 Architecture Aspects

In this section we will discuss the diversity mechanisms used for the families and how the platform architectures cover the five CAFCR views.

7.4.1 Platform Integration and Diversity

The main concept guiding the platform's use depends on the requirements specified for the platform and its architecture. If only a single system is needed, the architecture can describe a number of fixed functional blocks. If greater variations of a system are needed, this can be solved by for example introducing component frameworks (right picture in Figure 7-7). In the case of even greater diversity and flexibility a set of components can be used (component kit), from which the relevant components can be selected (left picture in Figure 7-7). A platform's integration level may lie between that of a complete system and those of individual components as illustrated in Figure 7-7.

Below, the software architecture of the three cases will be described, focusing on the support of diversity.

- The TSS architecture defines four horizontal layers. The layers contain components (called Building Blocks), based on a proprietary component model. A collection of component frameworks and plug-ins is used as the diversity mechanism. These component frameworks together constitute the system's architectural skeleton (right picture in Figure 7-7). More information on this architecture can be found in [50].

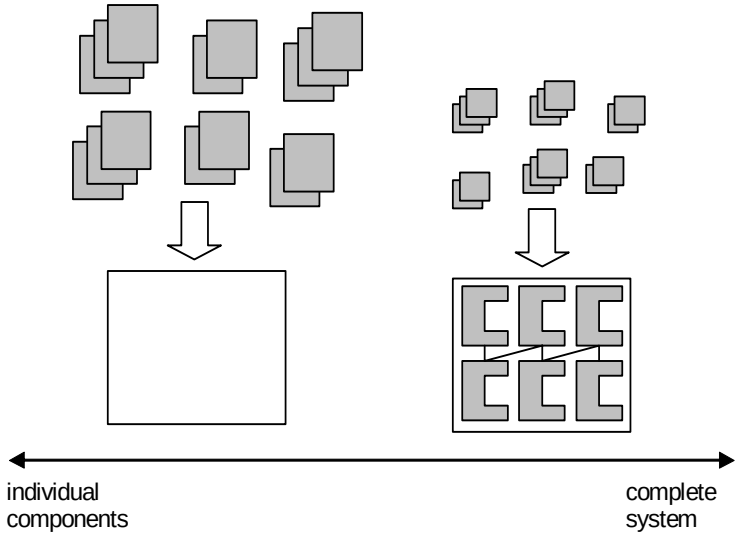

Figure 7-7 – Platform Integration and Diversity

Since this platform is already highly integrated, the platform architecture also constitutes the product architecture. As a consequence, a lot of documentation can be made in the context of the platform, which can be reused for the specific products. For example, a product is described as a list of features. These features are described as part of the platform documentation.

- The MMS architecture is somewhat similar to the TSS architecture. The system is also divided into a number of layers. Each layer contains a number of units, which can be regarded as kinds of subdomains, e.g. image processing, administration of patient data, etc. Each unit contains one or more components, realized with the aid of COM. Component frameworks and plug-ins are used as a diversity mechanism. More information on the architecture and variation mechanism for MMS can be found in [87] and [113].

  As in the case of TSS, this platform architecture also constitutes the product architecture. Product requirements documentation is handled as an extended subset of the set of requirements documentation provided by the platform.

  In the development of the MMS platform, it was not always easy to take all different architectural requirements into account. For some parts of the system, a component kit approach would have offered better flexibility.

- The MIS family architecture also defines a number of layers. In addition, a number of generic interfaces and information models are defined. These interfaces must be implemented by specific components, forming a platform. The platform's users must adhere to the family's architectural rules and guidelines, otherwise the platform components will not fit. More information on the MIS architecture can be found in [53].

7.4.2 Architecture View Coverage

In the TSS and MMS cases, the five CAFCR views have been reasonably covered by the platform from the start, e.g. analysis of customer needs, requirement specifications of products, realized component, etc. Although this is similar for both cases, the platform approach has been introduced in a different way. This is graphically shown in Figure 7-8, where the five rectangles represent the CAFCR views that are relevant for the platform, and the gray shapes the coverage of the first version of the platform. TSS was introduced with a product focus, but with a vision of a product family. Some product-specific parts fell outside the platform, and not all relevant parts of the platform were covered. This was dealt with in later releases (indicated by the arrows). MMS was introduced with platform focus with the platform boundaries being known, although not all platform functionality was integrated from the beginning.

For MIS, the introduction had a platform orientation. Furthermore, the initial focus was on the realized components (Realization view) and not as much on the earlier views of CAFCR. The idea is that this will grow over time (indicated by the two arrows in the right picture of Figure 7-8).

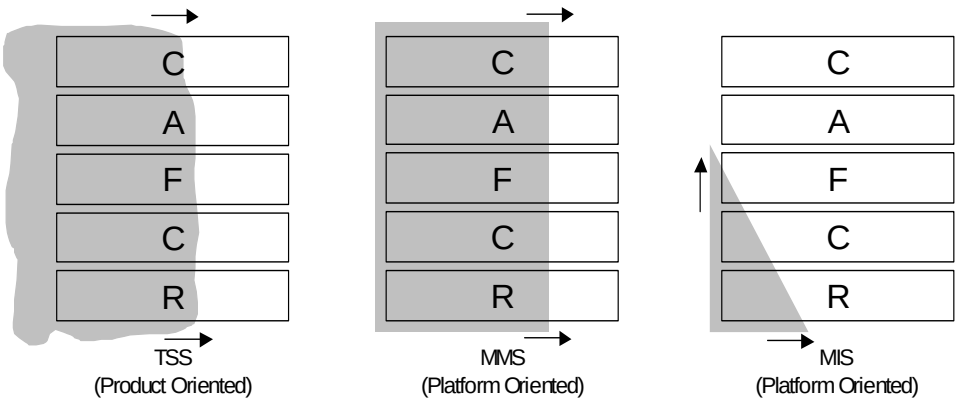

Figure 7-8 – Architecture Views and Introduction

7.4.3 Initial Stage

Some issues relating to the architecture that must be taken into account in the initial stage are:

- **Domain concept selection.** In the initial stage, the main concepts in the domain must be identified, e.g. subscriber and trunk lines or images and patient records. The stable and well-known concepts can already be introduced in the platform, so that they can be used for the development of generic interfaces or components. However, since the development of a platform has only just started, it will not be possible to define all relevant concepts. That is why the first step should not go all the way; evolution of the platform will always be necessary.
- **Guard conceptual integrity.** Important in the initial stage is to define the right architectural rules, guidelines, qualities, aspects, etc. Changing these architectural decisions afterwards will be hard and have a negative impact on conceptual integrity. Fortunately, these architectural decisions are usually stable and are not impacted too much by changes in domain concepts.

7.5 Process Aspects

In this section we will discuss some process structure and planning issues.

7.5.1 Process Structures

The three cases described here have different process structures. To enable comparison of these structures, we have mapped them onto the three activities described in [36]. These activities are:

- Application Family Engineering (AFE), which is responsible for the family architecture, ensuring reuse;
- Component System Engineering (CSE), which focuses on the development, maintenance and administration of reusable assets within the platform;
- Application System Engineering (ASE), which focuses on the development of products.

The processes of TSS can be mapped on these activities as shown in Figure 7-9. There are two types of processes, namely architecting and projects to create products. The architecting is a continuous process and is not related to a single

project. Most projects are defined to create one or more products. Sometimes a project is defined for an architectural redesign only. Several projects can run parallel to one another. A product project does not only deal with the product, but usually also contains some platform work, e.g. creating or modifying component frameworks.

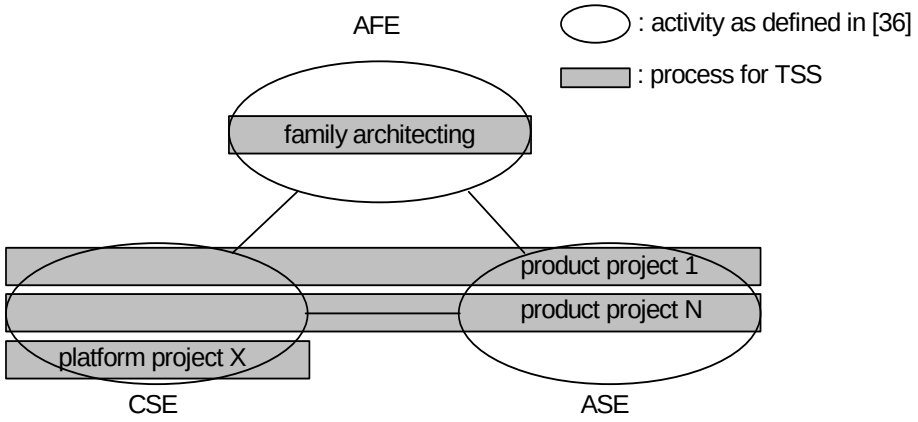

Figure 7-9 – TSS Process Structure

The MMS and MIS cases also involve an architecting process. But unlike TSS, there are separate projects, for working on the platform and on the products; see Figure 7-10. There will be only one project working on the platform at a time.

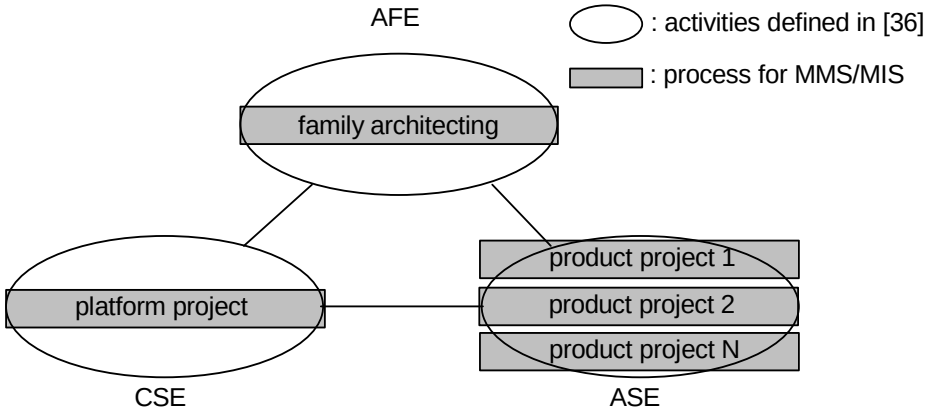

Figure 7-10 – MMS and MIS Process Structure

Each platform project of MMS focuses on a particular subset of platform users. The purpose is to extend the platform towards these users. Each MIS platform project focuses on adding new functionality to the platform. All interested

groups in the MIS family may use this new functionality. This difference in focus is related to the type of platform architecture that is applied. The MMS platform is based on architectural frameworks for the entire family, forming one skeleton. The possibilities of this skeleton must sometimes be fine-tuned with respect to the users. The MIS platform on the other hand is a component kit, to which new components can easily be added.

Below, some additional issues relating to the process structure will be discussed.

- **Family and platform architecting.** In the MIS case, the family architecture is broader than the platform architecture, as already discussed in the section on scoping. For MIS there are thus two architecting activities: the internal platform architecture, and the architecture with which the family members must comply. For the TSS and MMS cases, these two architectures are more or less the same.
- **Relation between CSE and ASE activities.** The TSS case differs from the other two cases in that there is a one-to-one relation between CSE and ASE activities. This gives a clear product focus within TSS. For the MMS case, this relation is one-to-few, i.e. the subset of products (about 3) on which the platform project focuses. For MIS this relation is one-to-many. The platform activity is further removed from the actual products. The difference in focus on platform or product work is shown in Figure 7-11 (derived from the HP Owen approach [105]). The MMS and MIS cases have a much more pronounced platform focus than the TSS case. This involves the risks of the architecture activity becoming too platform-focused and the product focus decreasing. If there is no clear product focus, the development goal may not be clear and this may lead to 'floating', i.e. the development becomes too generic and deadlines are missed. This can be tackled by increasing the cooperation between the CSE and ASE activities, as will be discussed in the section on organization. A risk involved in product orientation is pollution of the platform by 'fast hacks' that are not cleaned up afterwards.

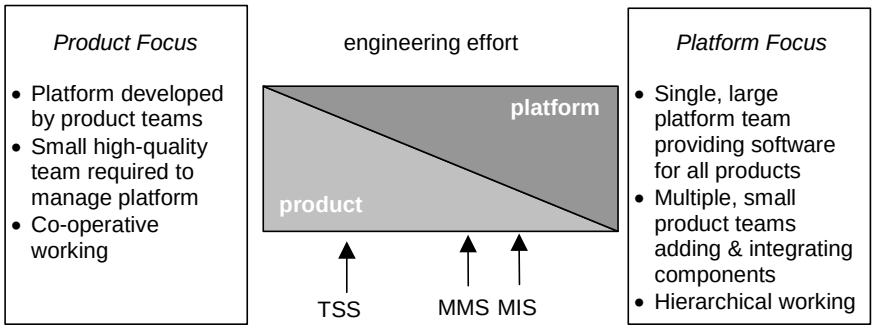

Figure 7-11 – Product vs. Platform Focus

- **Type of communication.** Of the three cases, TSS involves the smallest number of developers (taking into account both the platform and the product development). Furthermore, there is no clear separation between platform and product development; one developer can work on both a component framework and its specific plug-ins. The consequences of this are that the communication may be relatively informal and the amount of documentation required to transfer knowledge may be limited. The two other cases are more complex, and moreover entail clear separation between platform and product development. This requires more formal communication and elaborated documentation on how the platform can be used to build products. Extra attention must be paid to specifying clear interfaces. This has led to problems in the past: the documentation became too bulky or the product developers could not find the right information.
- **'Openness' of development.** Because of the nature of development at TSS, it is a kind of open-source development; a developer can modify both the sources of the component frameworks and the plug-ins in the platform. Such an open-source activity at a company is sometimes called inner source. This is a consequence of the combination of the CSE and ASE activities. The CSE activities of the MMS and MIS cases deliver black-box components; they should in principle not be modified by the product projects. Although MIS is currently in a closed-source mode, there are ideas to go in the direction of open-source. That would encourage contributions from the product builders to the platform.

7.5.2 Planning

At TSS, most projects are defined on the basis of products that have to be made. In these projects, the platform is also modified if necessary. For planning

purposes, a similar 'component topology' diagram as described for HP Owen [105] is used, in which it is indicated which platform components have to be updated or added to realize the new product. Several projects are performed per year, according to the customers' demands for specific products. Each project generates a set of compatible components, from which one or more products can be assembled. A set of compatible components is called a construction set. To limit the number of different construction sets existing side by side and to clean up 'fast hacks', projects are defined to clean up unnecessary diversity.

The MMS and MIS cases are both based on a heartbeat (release cycle) of one year, i.e. once a year a major release of the platform is made. In the meantime, smaller updates or bug fixes may be released. In the case of MMS, these heartbeats are each year related to a subset of the product in the family. In the case of MIS, the heartbeat is related to new functionality that is added to the platform. In both cases, roadmaps are made to indicate what can be expected which year. One of the reasons for this heartbeat is that new products are usually released once a year. The number of releases must also be limited so as to be able to test and control the set of components as a whole, although it would be possible to release individual components more frequently.

We have found that feedback is very important. This is especially true when platform development takes place separately from product development. At TSS, where platform and product development are merged, deadlines were not missed very often. Sometimes a 'fast hack' was introduced to meet a deadline, but it was then usually corrected afterwards. In the case of MMS and MIS, platform development has no real feedback from the market. A problem that occurred for example with MIS was that a lot of time was spent on defining generic interfaces, but they were not immediately realized in components. As a consequence, feedback from the implementation was missed. The interface specification could have been made much faster if feedback had been frequently provided. Feedback from the market is thus available for TSS more often (several releases per year) than for MMS and MIS (one release per year).

7.5.3 Initial Stage

Some issues relating to processes that must be taken into account in the initial stage are:

- **'Weight' of introduction.** During one of the sessions of the 4th Product Family Engineering workshop in Bilbao (October 2001), attention was paid to lightweight introduction of a product family approach. For example, Krueger [47] describes a family approach on code level that can be introduced without too much impact. Since several product

family initiatives did not succeed, lightweight introduction is now receiving more and more attention. Although the introduction itself is lightweight, it should from the very outset be based on a clear view of what the underlying architecture is otherwise there will be no sound basis.

Figure 7-12 presents the ideal approach, in which large investments are initially made (revolution), which are earned back later. This is however a risky undertaking, since this approach may fail because it costs too much and/or takes too long. The more realistic scenario that is sketched assumes partial, stepwise introduction without too much impact. This will lower the risk and result in benefits at an earlier stage. Feedback is obtained at an earlier stage this way, and acceptance can be built up gradually. However, the benefits may not increase as fast as in the ideal situation, because the approach will have to be improved gradually over time.

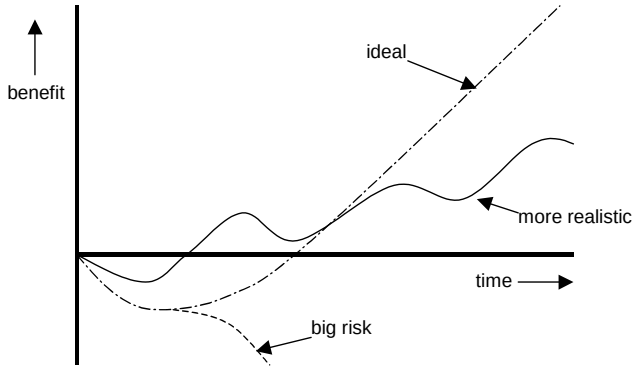

Figure 7-12 – Ideal vs. Realistic Introduction

The TSS platform was introduced gradually. The focus was on products. The various component frameworks were introduced (and redesigned) one by one. The MMS system, which involves an architectural approach similar to that followed with the TSS system, was introduced as a revolution. The MMS family, which had the high ambition of having a clean platform architecture from the start, had a longer introduction time than expected. Furthermore, with MMS we noticed that it is very hard to have the right requirements at the beginning and to identify all important concepts. The MIS system, based on a component-kit architectural approach, is being introduced step-by-step. Each release contains more functionality. This is made easy by the architectural approach of individual components. The focus

is initially on realized components, similar to Krueger [47]. Later, other views will also be taken into account; see the right picture in Figure 7-8.

- **Tool support.** Tools are needed to support the various processes. Tools are particularly important in the context of products based on a platform, because the platform will be used many times, allowing for automation of certain activities. For example, code for interfaces that are required to interact with the platform functionality can be generated. Another example consists of test specifications for generic interfaces that can be reused for specific components. The provision of such support must be planned at an early stage.

7.6 Organization Aspects

In this section we will discuss the organization structures and their mapping onto the process structures.

7.6.1 Organizational Structures

The organizational structure of TSS is depicted in Figure 7-13. There is one system team. This team is responsible for defining the requirements and the architecture. The architecture is designed and implemented by a number of development teams, one for each layer. All the teams work at the same site, and there is no separation between platform development and platform usage.

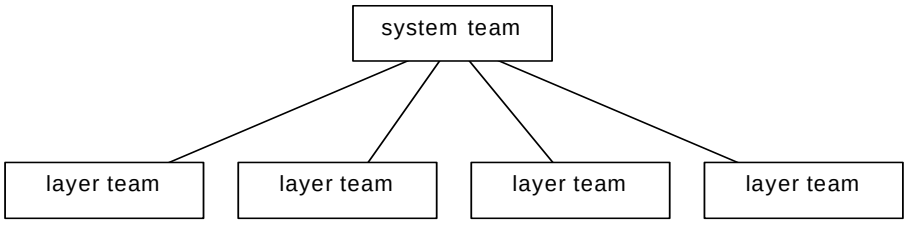

Figure 7-13 – TSS Organization Structure

In the case of MMS the situation is more complex. There is one platform team that is responsible for the platform requirements and architecture. The platform is designed and implemented by several teams, each of which is responsible for one or a few units. Some units are outsourced to a software house. The platform is used by a number of product groups, working in various countries. The management has to enable cooperation between platform development and platform usage.

Something similar holds for MIS as for MMS. One difference is that platform development is not divided into unit teams, due to the smaller size of the platform team. Outsourcing and cooperation with another company take place in the development of the components of the platform, too. Another difference is that there is an additional team that is responsible for the shared architecture for all members of the family. This team contains members from both platform-development and product groups.

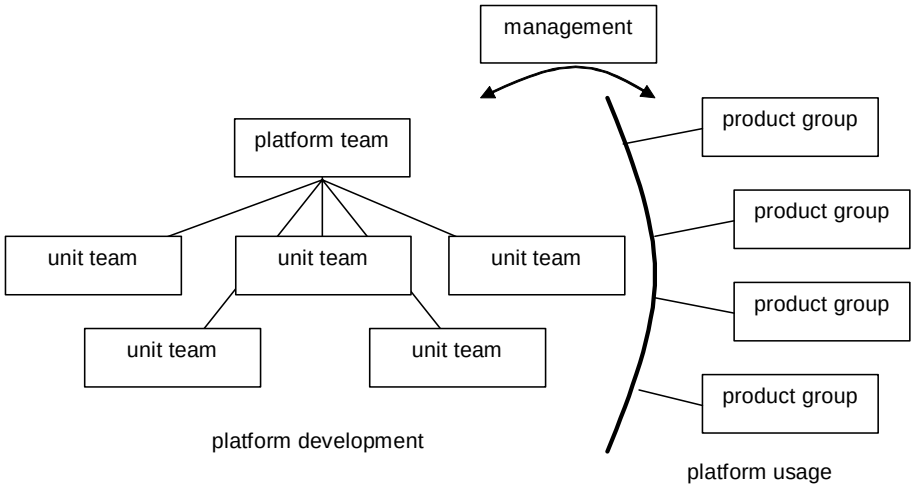

Figure 7-14 – MMS Organizational Structure

Comparison of Figure 7-13 and Figure 7-14 reveals a difference in the main decomposition of the organization. In the case of TSS, the main decomposition is along the layers (subdomains) of the architecture of the product family. Within such a layer team, work can be performed on several products in parallel. In the case of MMS and MIS, the main decomposition is into platform and product development. In [10] Bosch describes a number of organizational structures for product family development. The TSS case can be compared with the development department model. The difference is however that in the case of TSS the department is split into a number of teams, each dealing with their own subdomain, making this approach better scalable. The MMS and MIS cases can be compared with the domain engineering unit model. When the MIS and MMS cases are combined, this can be compared with the hierarchical domain engineering unit model.

In [105] it is assumed that when there is a separation into platform and product groups, the platform group is large and makes all the components, and the product groups are small and simply have to assemble the components into products (right-hand side of Figure 7-11). This does however not apply to the MMS and the MIS cases. The reason is that these platforms contain only the

shared components; specific functionality still has to be provided by the product groups. Furthermore, the MIS platform covers only one subdomain. This means that the product groups using MIS must be larger than the platform group.

7.6.2 Process to Organization Mapping

After the discussion of the processes in section 7.5, it is interesting to see how the processes map onto the organization. The easiest mapping would theoretically be one-to-one, but that is not always the case in practice. In the case of TSS, the system team is responsible for the architecting. The product projects run across the layer teams.

Unlike TSS, MMS and MIS introduce a separation between platform development and platform usage. This separation introduces a 'communication barrier' that has to be taken care of. For the MMS and MIS projects, the following solutions are applied:

- At MMS, members of the product groups (temporarily) participate in the platform development. This way, expertise from the product groups is used in the platform development and the members of the product group acquire knowledge of the platform. The architects and requirement engineers from the product groups are also involved in defining the platform's requirements and architecture.
- At MIS, there is a special team that is responsible for the shared architecture in which platform and product architects participate. Furthermore, when new functionality is added to the platform, this is usually done via a pilot project in which one or more product groups also participate.

Although the platform's components are currently made by the platform group, the product groups should in the future also be able to insert their components that can be reused by other groups into the platform. This comes close to the business unit model described in [10].

So, to improve communication between platform and product development, processes are needed that cross this organizational boundary. In Figure 7-15 this is graphically shown for the MIS case. In this figure the product groups also participate in the definition of the shared architecture and in writing components for the platform.

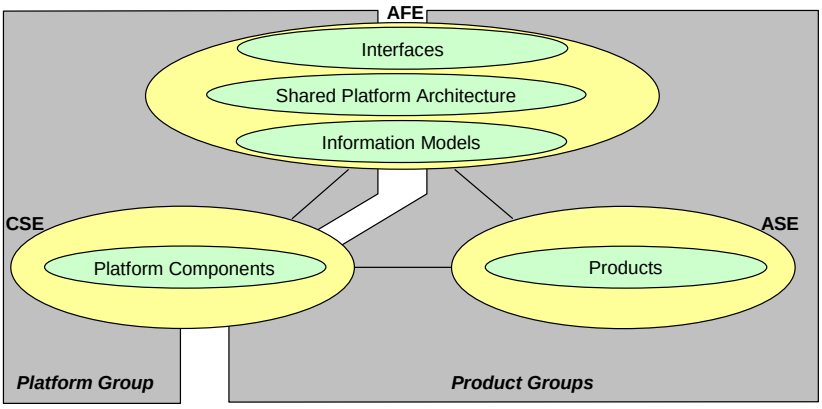

Figure 7-15 – Relation between MIS Processes and Organization

7.6.3 Initial Stage

Some issues relating to the organization that must be taken into account in the initial stage are:

- **Platform acceptance.** It is important to promote broad acceptance within the organization for a product family approach. It is especially important to create support within the product groups. These groups may see the introduction of a platform as a threat to their own freedom; they now have to adhere to the shared architecture. Since there are no separate product groups for TSS, this problem does not play a role in that context.

  In the case of MMS this is however more of a problem. Here, the product groups were used to define their own architecture and make their own decisions. This has changed, since the family prescribes an architecture that more or less corresponds to the product architecture. For the MIS platform this is less of a problem, since the MIS platform is only intended for a subdomain of the products, and a more flexible mechanism is used, i.e. the product groups are free to select the components they need. As a consequence, resistance will be much less than in the case of MMS; MIS is seen as something that can be useful to use, MMS is seen as something that must be used.

- **Distributed groups.** Both the MMS and the MIS cases have distributed development groups and apply outsourcing. This adds to the complexity involved in the development of a product family. As switching to a product family approach is already a very big step, involving a lot of

changes, any additional complexity should be minimized where possible in the initial stage. Feedback cycles must be short because a lot of decisions must be taken; outsourcing in this stage could imply less frequent feedback. If furthermore the groups are working in different countries, differences in country culture could complicate introduction.

- **Grow in size.** During the development of a new system (initial system stage) it is preferable for an organization to be small. The reason for this is that a number of issues have to be investigated and discussed before the system can be introduced, and this can be done more effectively in a small group. Later, when the requirements of the system, the software architecture, design concepts and principles are known and have been formulated, the number of people may increase (steady system stage). Only in this way can conceptual integrity be safeguarded from the outset.

7.7 Conclusions

In this paper we have discussed three product family approaches, each having its strengths and weaknesses. Product family approaches discussed in the literature usually focus on architectural issues. However, architectural issues are but a few of the many issues that have to be considered, as has been pointed out in this paper. To bring structure into these issues, we applied the BAPO reasoning framework. This helped us in comparing business, architectural, process and organizational aspects of the approaches. In Table 7-2 we have listed the most important issues in the form of questions that need to be answered when defining a product family approach.

This is only a subset of the issues that need to be defined for a product family approach. In this paper we have discussed some ways of dealing with these issues. For example, the problem of the 'communication barrier' introduced by the separation of platform and product groups can be dealt with by temporarily moving people between organizational units, or by involving members of different groups in pilot projects.

Special attention has been paid to the initial stage in which a product family approach is introduced in a company. This involves a considerable change effort, which must be well prepared. Some of the main problems that may be encountered are: underestimation of the change effort needed, wrong/incomplete requirements for the platform, wrong platform architecture, resistance within the company, restriction of freedom, no tool support.

Business	***product line - product family relation;*** What is the mapping between the product line (external) and the product family (internal) like? ***business integration;*** To what extent is the platform taken into account in the business planning? ***platform coverage;*** To what extent is the platform covered in the family? To a too great extent?
Architecture	***platform integration and diversity;*** What is the level of platform integration? Which architectural diversity mechanism is most suitable? ***architecture views coverage;*** Where lies the focus of the architecting e.g. realized components, family/product requirements, etc.?
Process	***family and platform architecting;*** Is the platform architecture the same as the family architecture? ***product or platform focus;*** What is the relation between the CSE and ASE activities; how is it organized? ***type of communication;*** How must the communication be organized, e.g. via documentation, exchange of persons, shared pilot projects, etc.? ***heartbeat and feedback;*** How often is the platform released; frequency of feedback?
Organization	***separation of platform and product groups;*** Must platform and product groups be separated (in relation to the size of the organization)? ***acceptance;*** How is the platform approach accepted in the various groups; how could this be improved?
Overall	***matching of BAPO aspects;*** Do the various business, architectural, process and organizational choices agree with one another? E.g. does the platform architecture conform to the business requirements? Are the process and organizational structure in agreement with the platform architecture?

Table 7-2 – BAPO Aspects for Product Families