Different Approaches used in Software Product Families

Rafia Inam

Mälardalens University.

Rafia.inam@mdh.se

Abstract

The use of software in consumer products is growing tremendously in current era. Further the complexity of software in products is growing, diversity increasing, and the lead time is decreasing. To meet all these challenges software reuse in consumer products is the solution. This evolves the concepts of software product family, product population, and product lines. Three different approaches are used to integrate all software within hardware. These three approaches are Integration-Oriented platform, Hierarchal and Composition-Oriented. The integration-oriented approach is a classical approach used for many years in industry but unable to meet the challenges of todays increased usage. Hierarchical and Composition-Oriented approaches are popular now-a-days to meet the challenges of industry.

1 Introduction

Now-a-days software is used a lot in consumer products. These consumer products have a lot of similarities in their interface and usage among product families. For example, take mobile phone as a product family; all mobile phones have the menus for making calls, storing contacts information, typing, sending, receiving and storing messages, etc. It will be very expensive if different software is developed for each kind of mobile phone.

Further the software for the products is becoming more and more complex, and diversity of the products is increasing rapidly. On the other hand, the lead time to develop the new products is decreasing. It means that the new products are more complex and diverse than previous ones and should be developed in less time. All this is only possible if software is reused in these products of same family (product families) and also in products of different families (product populations). This gives the idea of the reuse of software in product families, product lines and product populations.

At the start I describe the terms software product families, software product populations and software product lines in section 2. Section 3 gives the main challenges in the software development. Section 5 describes different approaches used to develop software product families in details with their advantages and disadvantages. A detailed comparison among these approaches is also illustrated in this section.

2 Background

This section describes the terms software product families, software product populations and software product lines.

2.1 Software Product Families

Software product families have gained much important from the increased usage of software in consumer products. "A software product family is commonly defined to consist of a common architecture, a set of reusable assets used in systematically producing individual products, and the set of products thus produced" [1]. One software product family normally has a very large number of products. The definition indicates that software components are reused on a common architecture because the products belonging to one family have a lot of common features and build upon a common architecture.

On one hand products in one particular family have lots of common features, but on the other hand they also have some differences too. Different methods are used to manage this variability in product families. These are described in second chapter of this survey.

An example given by R. van Ommering in [2] is of the families in Philips televisions. Following are different families:

- low-end televisions (e.g., for the bed room or camping),
- mid-range televisions,
- high-end televisions with a CRT as display,
- high-end projection televisions,
- high-end flat televisions (initially based on plasma displays), and
- Institutional televisions (for hotels, etc.).

All the above mentioned types are different from each other but when it comes to reuse of software, all are merged into a single family except the first one.

2.2 Software Product Populations

From the last decade, many different technologies are combined into a single product. For example; TV and VCR or TV and DVDs are combined, mobile phones are also providing features like photography, music, internet and GPS receiver, and radio and TV and much more. This convergence of different technologies presents the idea of product population.

R. van Ommering in [3] defines the term product population" for such a portfolio of product families, where products within a family have many commonalities and few differences, and where products between these families will have commonalities but also differences".

2.3 Software Product Lines

P. Clements and L. Northrop in [4] define the product line as "a software product line is a proactive and systematic approach for the development of software to create a variety of products". Hence it is a way to develop variability software for product families, which have not only numerous common features but also a great convergence of different technologies.

The three different approaches used for software product lines are Integration-Oriented approach, Hierarchical approach and Compositional approach. J. Bosch has discussed these approaches in details in [5, 6].

3 Challenges in Software Development

The enormous increase of use of software features in the consumer products from the last decade has put a big demand on industry to increase development of embedded software for product populations. Many researchers have worked to find the main trends behind this rapid growth. Some are discussed in this survey.

3.1 Complexity or End-to-End Solution

According to Van Ommering in [2] the software usage in consumer products is growing exponentially and it scales the Moore's Law. Fig. 1 from [2] shows the increasing trend of software during last decade. J. Bosch discusses more or less the same trend as end-to-end solution in [6].

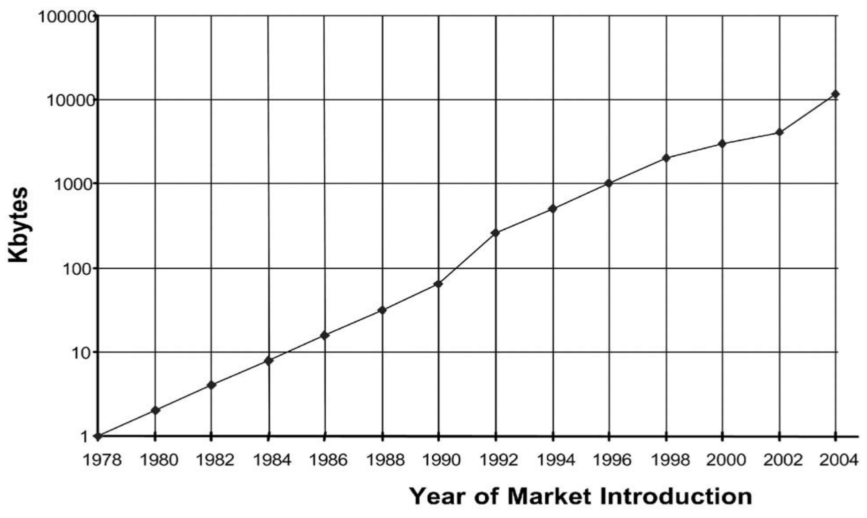


Fig. 1. Size of embedded software in high-end televisions. [2]

With the increase in software, its complexity also increases. Initially functionality was mainly embedded within the hardware for products. But now with the availability of fast processors and big memory storages, it is more feasible to implement it in software. Further many features that are attractive to the user and that give products more sophisticated and professional look can also be implemented in software.

3.2 Convergence or Diversity

According to Bosch [5, 6], from the last decade convergence of electronic product is continuously taking place with the IT and telecommunications fields. For example mobile phone is also using photography technology; TVs are connected to Internet via TCP/IP; and PDAs are coming with GSM and GPS receivers. This convergence results in the growing diversity of the consumer products. Fig. 2 illustrates the divergence aspect in Philips TVs by Van Ommering in [2].

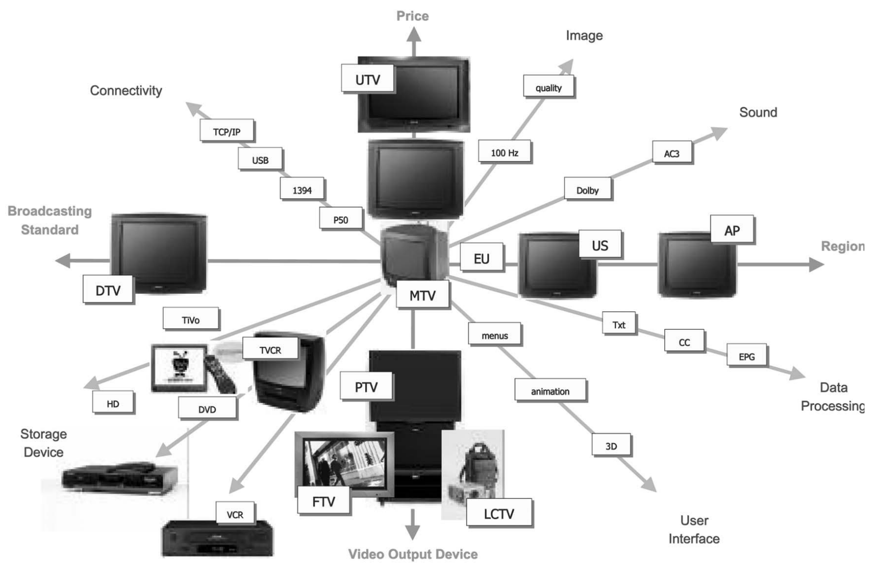


Fig. 2. Axes of diversity [2].

3.3 Lead-Time

Lead time shows the time in which a new product should come into the market. This is very important for companies to get a good business in market. Traditionally companies deploy new products on annual events like Christmas or on big sports events. But now this lead time is decreased to 6 months [2].

3.4 Software Engineering Capability

Hence to meet this lead time, the company must have good capabilities to develop complex software in this short time [5]. Further this software must also implement the divergence required for that product.

3.5 Summary

Summarizing all these above point, we conclude that the companies are facing the challenges to build complex software with all diversity implemented into products and to develop these products in less lead time. But at the same time, to get a good profit from the market, companies should also maintain the cost of the products within the customer range.

If the software is developed from the scratch for each product then clearly the company cannot meet the lead time and the cost of product will also become out of range of the customer. Hence product families, product lines and product populations have become an integral part of the industry now-a-days. According to Bosch [5] due to all these trends "the organizations are stretching the scope of their product families far beyond the initial design".

4 Different Approaches used in Software Product Families and their Comparison

The companies that are using software product family approach are very successful in market and in their business. As a consequence of this success these companies are now "stretching their product families significantly beyond their initial scope" [5, 6]. Companies are not only including more and more new products into the families but also adding their previous products into the family. Ommering gives an example in [2] of Philips Company where mid-range televisions are added into the product family that was not initially into that family. Bosch says that "this easily causes a situation where the software product family becomes a victim of its own success" [5, 6].

Broadening the scope of the software product family, with increased complexity and diversity, and to deliver the product family in due time is a real challenge. The traditionally used Integration-Oriented platform approach to develop software product families is becoming insufficient to adopt increasing scope, complexity, and diversity of the product families. Hence some alternative approaches are required to overcome this intricacy. Bosch has given two alternative approaches in [5, 6] that are Hierarchical Software Product Family and Composition-Oriented approach. Before discussing those two approaches, we are describing the Integration-Oriented Platform approach and some of its deficiencies in expansion of the scope of families.

4.1 Integration-Oriented Platform Approach

Integration-Oriented Platform approach is a traditional approach used by many companies to develop software product families. In past companies has been very successful in business and market by using this approach [5, 6]. The famous Hewlett-Packard (HP) company used this approach for many years in their printers and all-in-one devices [7]. This approach is very well-suited in situations where the family has a small scope and all the products have related features. The problem in this approach begins when the scope of the family has to expand. The company may want to add previous unrelated products into the family or to add additional features into its products.

Here we describe the basic procedure of Integration-Oriented Platform and problems in this approach. This approach works in two cycles: Platform organization cycle and product organization cycle.

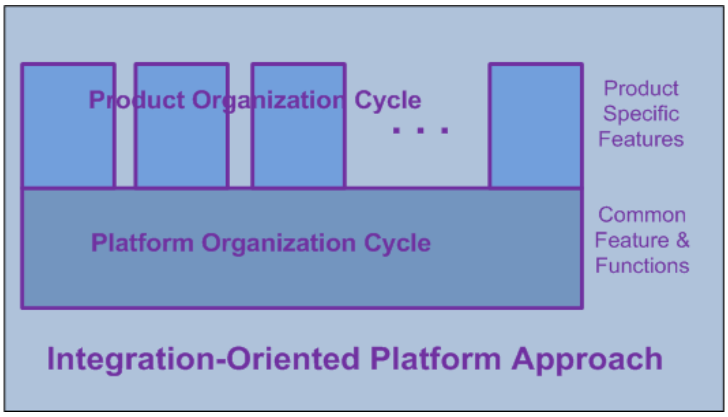

Product Organization Cycle
. . .
Product Specific Features
Platform Organization Cycle
Common Feature & Functions
Integration-Oriented Platform Approach

Fig. 3. Two cycles of Integration-Oriented Platform Approach

4.1.1 Platform organization cycle

All the common functionality, features of a product family and the architecture are implemented in a platform. It provides the basis for the development of product-specific features in the next cycle. Normally there are many teams working on platform organization development. Each team is responsible for a particular component (or a set of related components) development. All these components are then integrated regularly into the platform and tested and validated. It is a periodic release cycle and the cycle duration is quite large.

4.1.2 Product organization cycle

This cycle takes the platform as its basis in which all the common features and functionality has been added, integrated, and tested and verified for correctness. Product organization teams then implement the product-specific features and finalize their products. Cycle duration is less than the Platform organization cycle.

4.1.3 Problems in Broadening the Scope of Product Families

This Integration-Oriented Platform approach has been very successful in past when product families were relatively small but now-a-days the scope of the product families is growing a lot and this approach has become insufficient to add a lot of new products into the family in an efficient way.

This approach is too much centralized and a lot of communication and good understanding is required between the platform and product development teams. As a large number of features are incorporated in the platform, the platform development cycle is bigger than the product organization cycle. Therefore, platform development team has the pressure to complete platform and they can use shortcuts to develop that effects quality. And sometimes the integration, testing and validations are shifted to integration team that cannot understand the exact origin of problems. Some major problems described by Bosch [5, 6] are given below:

1. Decreasing complete commonality
2. Increasing partial commonality
3. Over-engineered architecture
4. Cross-cutting features
5. Maturity of product categories
6. Unresponsiveness of platform

4.2 Hierarchical Approach

When the software product family evolves, a large number of reusable components are developed. One way to organize these reusable components is to arrange them in a hierarchical fashion. These software reusable components and the architecture of the product family are organized in three different layers. All the basic and common functions and features of the software product family are implemented in Basic Platform Layer. Second layer above it is Reusable Business Unit Layer that is developed over basic platform layer. It consists of some extensions to the architecture and a set of reusable components that are common to a particular family. The top layer is reusable product family and developed over the business layer. The development of new products takes place here and using this hierarchical approach it is very fast and easy.

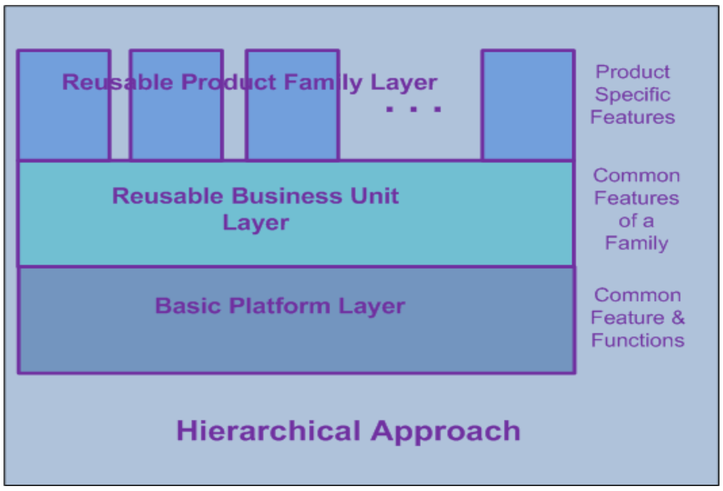

Reusable Product Family Layer
. . .
Product Specific Features
Reusable Business Unit Layer
Common Features of a Family
Basic Platform Layer
Common Feature & Functions
Hierarchical Approach

Fig. 4. Three Layers of Hierarchical Approach

4.3 Composition-Oriented Approach

This approach is entirely different from the first two approaches as it has no common platform with common and basic functions and features implemented in it. Instead all the basic and advances features are developed into the reusable components of a software product family. To develop products these reusable components are combined with each other through well-defined interfaces. Hence the architecture of the product family is implemented within the reusable components. An example is given in Fig. 5. Van Ommering comments it as “this approach lies between classical decomposition approach and opportunistic use of commercially available components COTS, thus forms the bridge between the software product line and the COTS community” [2].

4.3.1 The Koala Component Model

Koala is an example of Composition-Oriented Approach given by Van Ommering [2]. It is used to develop software product families in Philips TVs.

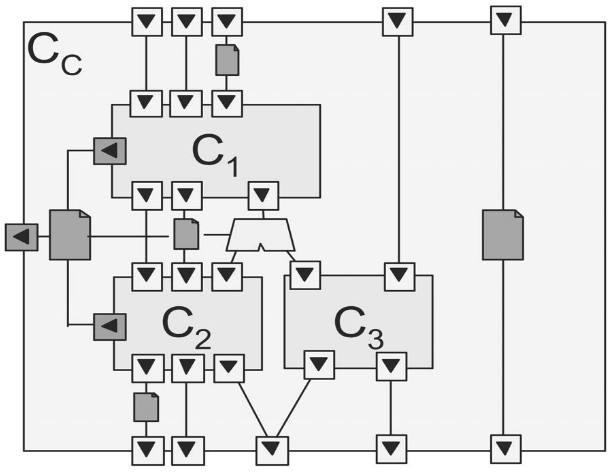


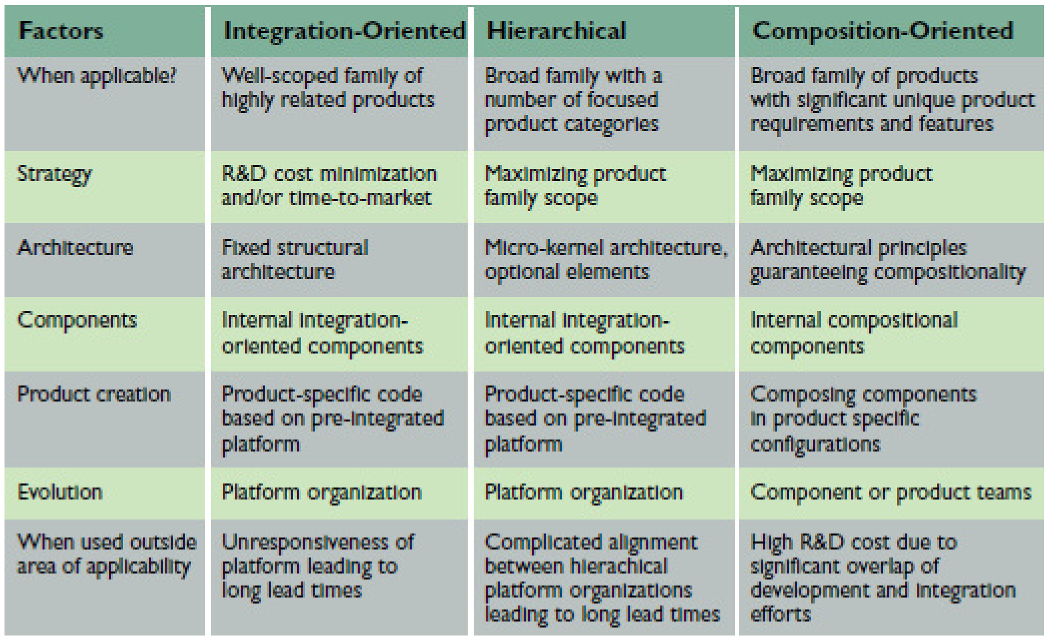
Fig. 5. Example of Koala components [2].

“Koala is an architectural description language (ADL), and one of the few designed to handle diversity” [8]. It is widely used is industry. The components in Koala are integrated with each other using interfaces. These components can have two different types of interfaces. First is Provides interface. Functionality is provided to the environment by this interface. When functionality is required from the environment then Requires interface is used. These interfaces can be connected with each other either directly or sometimes they are connected through some code in the middle. This code is called the glue code, code used to connect two interfaces. In fig. 5 the “document shaped objects” represent glue code. Sometimes this glue code is already defined; it is called a switch in this situation. In fig. 5 a “short pair of trousers” represents a switch.

4.4 Comparison of three software product family approaches

Jan Bosch discusses the comparison among the three software product family approaches in details in [5, 6]. The factors on which this discussion is made are applicability, business strategy, architecture used, components, product creation, evolution, and use outside the applicability area.

1. First of all the applicability area of each approach is given. Integration-Oriented approach can only be used in a small scoped family in which all the products are highly related. Using Hierarchical approach the scope of family can be broadens, but only a number of focused products can be added. In Composition-Oriented approach, the scope can be broadens and the products with unique functions and features can be added very successfully.



Factors	Integration-Oriented	Hierarchical	Composition-Oriented
When applicable?	Well-scoped family of highly related products	Broad family with a number of focused product categories	Broad family of products with significant unique product requirements and features
Strategy	R&D cost minimization and/or time-to-market	Maximizing product family scope	Maximizing product family scope
Architecture	Fixed structural architecture	Micro-kernel architecture, optional elements	Architectural principles guaranteeing compositionality
Components	Internal integration-oriented components	Internal integration-oriented components	Internal compositional components
Product creation	Product-specific code based on pre-integrated platform	Product-specific code based on pre-integrated platform	Composing components in product specific configurations
Evolution	Platform organization	Platform organization	Component or product teams
When used outside area of applicability	Unresponsiveness of platform leading to long lead times	Complicated allignment between hierachical platform organizations leading to long lead times	High R&D cost due to significant overlap of development and integration efforts

Fig. 6. Comparison of three software product family approaches [5, 6].

2. Second point discusses the business strategy used in these approaches. Using Integration-Oriented platform approach R&D cost is greatly reduced become software is shared among a number of products. This approach also meets the Time-to-market or lead-time because of sharing/reuse of software. This approach however does not help in broadening the scope of the family. While Hierarchical and Composition-Oriented approaches not only support time-

to-market and R&D cost reduction, they also work well when maximizing the scope of family.

3. Third point is the architecture used by these three approaches. Integration-Oriented approach uses a fixed structural architecture which provides a complete basis for product family with a little variation points. Hence it is very difficult to enhance the scope of family as the complete architecture has to be altered in this case. Hierarchical approach uses Micro-kernel architecture that consists of micro-kernels and some optional elements. These optional elements can be changed, new elements can be included or some elements can be deleted. In this way this approach gives more freedom to add or change the product family. Composition-Oriented approach uses architectural principles guaranteeing compositionality which has no separate layer for architecture. Instead all architectural principles are added into the components. Thus architecture can be changed very easily in this approach as compared to other two approaches.
4. Although architecture provides the basis, the actual functionality and requirements are added into components. Integration-Oriented and Hierarchical approaches use internal integration-oriented components in which components have variation points but still they are very much dependent on other components. Composition-Oriented approach uses Internal compositional components in which components are based on well-defined architectural principles. These components are relatively independent and can be freely composed with each other.
5. Product creation defines how the products are developed. Integration-Oriented and Hierarchical approaches both use product-specific code based on pre-integrated platform in which all the basic generic functionality has been implemented into pre-integration platform. Products are developed at top of the platform. Composition-oriented approach composing components in product specific configurations in which components are interconnected with each other.
6. Companies can take advantage of reusable components if all products are created and then continuously evolved over time. Integration-Oriented and Hierarchal approaches use platform organization as basis. For evaluation, all the new features and requirements should be pre-integrated into the platform. It is more time consuming and costly. In Composition-Oriented approach evolution takes place using extending the components. New components can be added or old can be deleted easily. Thus evolution is relatively easy in Composition-Oriented approach.

5 Summary

Software product families, product lines are the solution to meet the challenges in software development. Different approaches have been used to develop software for consumer products like Integration-Oriented platform approach, Hierarchal approach and Composition-Oriented approach. Observing all the advantages and disadvantages of these approaches, I can conclude that the Composition-Oriented approach can meet all the requirements and challenges in a better way than the other two approaches.

6 References

[1] J. Bosch," Design and Use of Software Architectures: Adapting and Evolving a Product-Line Approach", Addison-Wesley, Boston, MA, 2000.

[2] R. V. Ommering, "Software reuse in product populations", IEEE Transactions on Software Engineering, Volume: 31 Issue: 7, July. 2005, Page(s): 537- 550.

[3] R. V. Ommering, "Building Product Populations with Software Components," In Proceedings of Int'l Conference on Software Engineering, pp. 255-265, May 2002.

[4] P. Clements, L. Northrop, "Software Product Lines, Practices and Patterns". Addison-Wesley, 2002.

[5] J. Bosch, "The Challenges Of Broadening The Scope Of Software Product Families", Communications of ACM, Dec 2006, Vol. 49, No. 12.

[6] J. Bosch, "Invited Talk: Expanding Software Product Families: From Integration to Composition", ARCS 2007, LNCS 4415.

[7] P. Toft, D. Coleman, J.Ohta, HP product generation consulting, a cooperative model for cross-divisional product development for a software product line. In P. Donohoe, Ed., *Proceedings of the First Software Product Lines Converence (SPLC1)*, Kluwer Academic Publishers, 2000, 111–132.

[8] N. Medidovic, R.N. Tayler, "A Classification and Comparison Framework for Software Architecture Description Languages," IEEE Trans. Software Eng., vol. 26, no. 1, pp. 70-93, Jan. 2000.

[9] Ivica Crnkovic, Magnus Larsson, "Building Reliable Component-Based systems", Artech House Boston. London. 2002.