TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

COMPARISON OF VARIABILITY MODELING TECHNIQUES

Qammer Abbas

Asif Akram

MASTER THESIS 2009
INFORMATICS

TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

COMPARISON OF VARIABILITY MODELING TECHNIQUES

Qammer Abbas

Asif Akram

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom ämnesområdet informatik. Arbetet är ett led i masterutbildningen med inriktning informationsteknik och management. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Kurt Sandkhul
Examinator: Vladimir Tarasov

Omfattning: 30 poäng (D-nivå)

Datum: June 15, 2009

Arkiveringsnummer:

Postadress: Box 1026 551 11 Jönköping | Besöksadress: Gjuterigatan 5 | Telefon: 036-10 10 00 (vx)

Abstract

Variability in complex systems offering rich set of features is a serious challenge to their users in term of flexibility with many possible variants for different application contexts and maintainability. During the long period of time, much effort has been made to deal with these issues. An effort in this regard is developing and implementing different variability modeling techniques.

This thesis argues the explanation of three modeling techniques named configurable components, feature models and function-means trees. The main contribution to the research includes:

- A comparison of above mentioned variability modeling techniques in a systematic way,
- An attempt to find the integration possibilities of these modeling techniques based on literature review, case studies, comparison, discussions, and brainstorming.

The comparison is based on three case studies each of which is implemented in all above mentioned three modeling techniques and a set of generic aspects of these techniques which are further divided into characteristics. At the end, a comprehensive discussion on the comparison is presented and in final section some integration possibility are proposed on the basis of case studies, characteristics, commonalities and experience gained through the implementation of case studies and literature review.

Sammanfattning

Variation i komplexa system som erbjuder rika uppsättning av funktioner är en allvarlig utmaning mot sina användare i form av flexibilitet med många varianter för olika tillämpning sammanhang och underhåll. Under lång tid mycket ansträngningar har gjorts för att hantera dessa frågor. En insats i detta avseende är att utveckla och genomföra olika variationer modelleringsteknik.
Denna uppsats hävdar att förklara tre modelleringsteknik namngivna konfigurerbara komponenter, funktion modeller och funktioner betyder träd.

De viktigaste bidrag till den forskning som omfattar:

- En jämförelse av ovan nämnda variationer modelleringsteknik på ett systematiskt sätt,

- Ett försök att hitta den integration möjligheterna av dessa modeller bygger på litteraturgenomgång, fallstudier, jämförelser, diskussioner och brainstorming.

Jämförelsen är baserad på tre fallstudier som var och en har genomförts i alla ovan nämnda tre modelleringsteknik och en uppsättning generiska aspekter av dessa tekniker som delas upp i egenskaper. Efter en omfattande diskussion om jämförelsen presenteras och i sista avsnittet några integration möjligheten föreslås på grundval av fallstudier, egenskaper, gemensamhet och erfarenheterna från genomförandet av fallstudier och litteraturgenomgång.

Acknowledgements

The master thesis has been carried out at School of Engineering at Jönköping University. The road to the thesis spans over several months of literature review and discussions with supervisors, whenever needed. The authors acknowledge the contribution of those who have helped for the completion of thesis.

In particular, we wish to express our gratitude to our supervisor, Professor Kurt Sandkuhl for his guidance and encouragement to choose the topic and start our work. His invaluable suggestions and support through out the work has helped us in many aspects of organizing and conducting the research.

We are also deeply indebted to our external supervisor Christer Thörn for sharing valuable suggestion at times during the meetings and his insightful view of our work lead to the completion of work.

Furthermore, we would also like to include our gratitude to Vladimir Tarasov, Program supervisor who has enabled this research work.

Finally, we want to thank our families for the encouragement and support. A special thought is devoted to our parents for a never-ending support and love.

Key words

Feature modeling, function-means tree, configurable components, comparison, variability, integration possibilities, modeling techniques

Contents

List of Figures

List of Abbreviations

ODM	Organization domain modeling
STARS	Software Technology for Adaptable Reliable Systems
CARDS	Central Archive for Reusable Defense Software
SEI	Software Engineering Institute
SPL	Software Product Lines
FODA	Feature-Oriented Domain Analysis
FORM	Feature-Oriented Reuse Method
QLE	Quality, lead-time, and effective/efficient
QFD	Quality function deployment
FR	Functional requirement
DP	Design parameter
C	Constraint
CC	Configurable components

1 Introduction

Systems offering a broad set of features to their users are very complex and cause serious challenge regarding flexibility and maintainability to their developers and managers. Actually, there are so many variants of the system modules and application contexts, that it is very hard to achieve high flexibility of a system along with maintainability and manageability at the same time.

For example, in the automotive car industry the number of electric and electronic car sub-systems has tripled during the last 20 years. Most of these systems are available with many variants to meet requirements from different countries or to fit to different models of a car manufacture.

A typical case is German car maker BMW which indicated that only 2 cars in two million produced at BMW have exactly the same configuration of subsystems, special equipment or optional parts [16]. Similar is the case for software product lines where more and more lines of codes with different functionality are being added on daily basis.

Variability modeling is not only constrained to mechanics rather it has also been addressed in software industry and service oriented systems. For example, in software industry there are issues related to software product lines, that is, reusable components, alternative modules, interface variations in different versions etc. As long as an organisation develops it processes and gain experience and knowledge, in order to get profit of that knowledge, it needs a way to capture and handle variability in its domain analysis and domain engineering processes.

Handling of all these issues is called Variability Management. Variability management is achieved through variability modeling, and then using it for decision making. There are different variability modeling techniques such as feature modeling, function-means trees, and configurable components in addition to some other approaches. All these variability modeling techniques have their specific area of application, specific set of notations and specific way of implementation where they provide the best suitable solution.

1.1 Background

Variability modeling techniques are solutions to manage variability in product families. These techniques include feature modeling, function-means tree and configurable components among others. All of these techniques have the typical area of application, modeling methods and semantics. The implementation of one technique in a particular area is more useful than the others. Also these techniques have their own pros and cons. But what these pros and cons exactly are and what are the commonalities and differences between these techniques, which benefits could be achieved by combining these techniques and how the integration should be done. All these questions are very important, and if answered, is certainly a worth full contribution in the area of variability modeling. So there is a need to compare these techniques and then, if possible, to integrate these into a single technique where all of the benefits of these techniques can be achieved.

1.2 Purpose/Objectives

The purpose of this thesis is to compare and analyze the commonalities and differences among variability modeling techniques and then propose integration possibilities, so that a better approach can be adopted while managing and maintaining variability.

1.3 Limitations

Scope of our thesis is limited to comparison and proposing the integration possibilities and not to give a framework to integrate due to limitation of time and expected area of research.

1.4 Thesis outline

The next part of the thesis presents the theoretical background and review of the literature for three selected variability modeling techniques followed by research methodology that defines how the research in this thesis has been carried out. After that, a comparison framework definition is outlined which includes characteristics defining what they really meant in the context of this research. The implementation of three case studies in the selected variability modeling techniques with figures and explanation comes after that. Next is comparison on the basis of characteristics, case studies and theories available in literature. Finally, results and integration possibilities are discussed in the light of literature and experience gained during the research work.

2 Theoretical Background

Variability is the variation and commonalities among different parts, aspects or features of a product family. These variations can be seen from various perspectives in an organisation or in a product family, for example from customer requirements perspective where a particular instance of a product family may or may not have a special feature in it, and also from an engineer's point of view where he/she sees the product as a set of parts in a hierarchy, and then derivation of the product by selecting different compulsory, alternative or optional parts.

Dealing with this kind of variability is a problem when number of product of family increases, here it's very hard to manage and keep track of all of the variations of a product family. The organization gets experience by developing new products or products variations by time and matures its design and development processes. This experience is very important and, in order to get benefit from, it must be utilized and reused. On the other hand, sometimes if poorly managed it may become counterproductive because there will be a lot of information which is redundant and scattered around the process of development and will definitely lead to wastage of resources and require more effort in order to acquire the correct information when needed.

Variability modeling resolves this problem to a certain point if properly implemented. It is the key to variability management and plays an important role in an organization especially during the product derivation phase. It is used to model variable aspects of a member of a product family. Instances of this model are then usually supposed to be used to derive new products of that family efficiently. Over a few decades, several variability modeling techniques are developed out of which feature modeling, function-means tree and configurable components are of the great importance. In the following sections, comprehensive information and theoretical background with detailed descriptions of the working of these techniques is given.

2.1 Feature Modeling

"The purpose of feature model is to extract, structure and visualize the commonality and variability of a domain and a set of products", say Sandkuhl and Thörn in their paper [1]. Feature modeling basically deals with capturing the common and variable features of a product family and provides a way to model and visualize it through the defined notations presented as diagrams. Variability is the differentiation among the features of the products which provides options to satisfy customer requirements. But this is not the only aspect which variability modeling covers, it also deals with other aspects regarding domain analysis and domain variability modeling. This is a hierarchical modeling where variability and commonality is modelled as different features and put into a hierarchy of feature and sub features. This hierarchy of features is modelled and visualised through the features diagram. The example of a general feature diagram is presented in the figure 1.

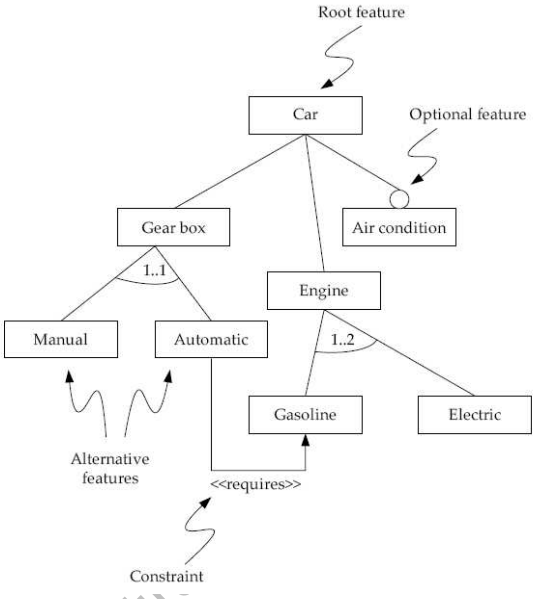

Figure 1: Example of partial feature model of car [1]

In figure 1, structure variability of the car is modeled and presented as features tree where a root feature is car. Gear box, engine and air condition are sub features of car, of which air condition is shown as optional feature with a circle at the end of edge. We can also see that there is a notation to present optional features and their multiplicity too. The multiplicity here is important and it defines number of features to be selected among available set of optional features. This is shown with two bound values, e.g. 1..2 means at least one or maximum two feature can be selected at a time in a particular configuration of the product. Intra-feature relationships are shown externally by connecting two features and putting some appropriate name on the relation. But this one is very general example we will discuss and present more notations about how to show different connections in hierarchy in the following sections.

2.1.1 History of Feature Modeling

As for as software industry got started evolving , work on standardizing its process, and the need for reuse of the experience and design was more and more. Different procedures were adopted to achieve the reuse like domain engineering. Domain engineering plays a basic role in the evolution of feature modeling. The aim of domain engineering is to facilitate the organized software reuse by modeling the knowledge, expertise and capabilities within the business area of an organization. The intention of modeling a domain is that when an organization which has constructed and conducted its business, it gathers the knowledge and experience of that area. As this knowledge is from the same business area where the organization will probably work in the future developments, and the systems constructed in the future will probably share same technical characteristics and will have to meet the similar requirements, so it is likely that organization can benefit from the acquired knowledge and thus can save from reinventing the wheel.

The idea of domain analysis was introduced in the work conducted on software families in the mid 1970's. An overview of FODA, ODM, STARS, CARDS and SPL is obtained from [3][4][5][6][7], and is inspired by [1],The term domain analysis as described by Neighbors [2] is "the activity of identifying the objects and operations of a class of similar systems in a particular problem domain". Later several methods to perform the domain analysis were developed by the time, among those some important and popular methods are Feature-Oriented Domain Analysis [3] and Organization Domain Modeling (ODM) [4]. Software Technology for Adaptable Reliable Systems (STARS) [5] was a project funded by U.S department of defense which conducted a lot of research on domain engineering and domain based reuse of software and design. One of the sub projects of these was Central Archive for Reusable Defense Software (CARDS) [6]. These projects ran through several years, stages, directions and institutes. As a result, Software Engineering Institute's (SEI) Software Product Lines (SPL) [7], ODM and some other methodologies and guidelines for domain analysis were developed.

Neighbors gave his idea of domain oriented reuse as "it seems that the key to reusable software is to reuse analysis and design; not code." Later some others refined the idea of what to be captured as the result of domain analysis and proposed that domain model could be equipped with more advanced concepts like use cases, feature models and concept models like class diagrams etc [8].

Software product line (SPL) is a methodology that uses the domain engineering and Software reuse principles [7]. While SPL uses almost the same steps involved in other domain engineering methodologies but it puts more emphasize on the management of activities good coordination and supervision of the phases involved in the domain engineering. It proposes the iterative nature of activities and communication among different ongoing processes in an organization.

SPL work flow consists of three activities. Core asset development together with management constitutes domain engineering, whereas product development and management constitute application engineering.

Feature modeling as a concept first arrived in a report by Kang et al. in a report describing FODA (Feature-Oriented Domain Analysis) [3]. The proposed use of feature models was to facilitate the domain analysis and later, when it was started being used in other domains too, it started extending the original definitions, notations and concepts used by FODA.

FODA was followed by the successor FORM (Feature-Oriented Reuse Method) [9]. Then a major work was done by the Czarnecki and Eisenecker in Generative Programming regarding formalizing the features models [10] and further refinement of the feature model notation and diagrams was published by Riebisch et al [11, 12]. Though it can be argued that feature modeling can be older than FODA but it was FODA which at the first time introduced the structured feature modeling technique.

Feature models were originally used in domain analysis but with the passage of time it evolved its notations and domain of products and now is an effective way of representing the commonalities and variability. Now a day, it is mainly used in representing common requirements and for configuration and automated construction and instantiation of from the product line described by the model [17].

2.1.2 Feature Types

Features may be categorized as concrete, abstract/pseudo or parameterized. Concrete features are those which can be realized and actually implemented, while abstract feature facilitate only the building structure of the product. Parameterized features are assigned a value if included [1].

Different types of features and notations are available in literature of which most accepted and important are described below:

Mandatory –

These types of feature are required to be present in every type of product where their parent feature is selected, these type of feature include the functionality which is mandatory for every type of product configuration of the product under development or in process of modeling.

Optional –

These features represent the variability of the product under consideration and it may or may not be included in the product.

Alternative –

These are the features which are to be included exclusively out of a set of available features, that is, one out of the available features should be included but not all. These features, represent the variation is the product configuration at hand. Variation possibilities in the product line is directly proportional to number of features, that is, more number of features means more variation possibilities.

2.1.3 Feature Diagrams

Feature diagrams are hierarchical decomposition of feature models showing common relations which include dependencies, commonality variability constraints. A feature diagram is usually visualized as tree structure with a root node representing the more general concept and then sub nodes in the tree representing the sub concepts as shown in the figure 2.

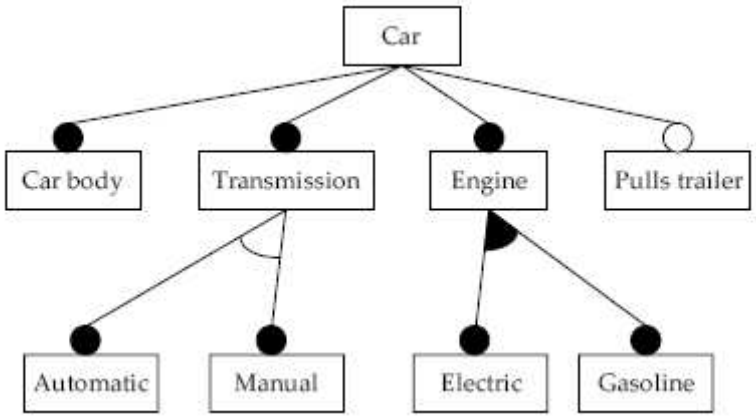

Figure 2: Example feature diagram with notations [1]

Notations of Diagram –

At first, the notations in feature diagrams come from [3], these contains the basic notations like mandatory, optional and alternative features and rules like dependency and mutual exclusiveness.

Later, layered model was proposed by the successor of FODA, called FORM. This introduced four layers in feature diagram named as capability layer, operating environment layer, domain technology layer and implementation techniques layer.

Today, in feature modeling notations for mandatory, alternative, optional, and parameterized features are mostly used. To represent mandatory features filled circles are used, for optional features empty circles are used while for alternative features arcs are used with multiplicity. This can be seen in the figure 3 as described in [11]. Later while during the implementation of case studies we will use these notations for building feature model diagrams.

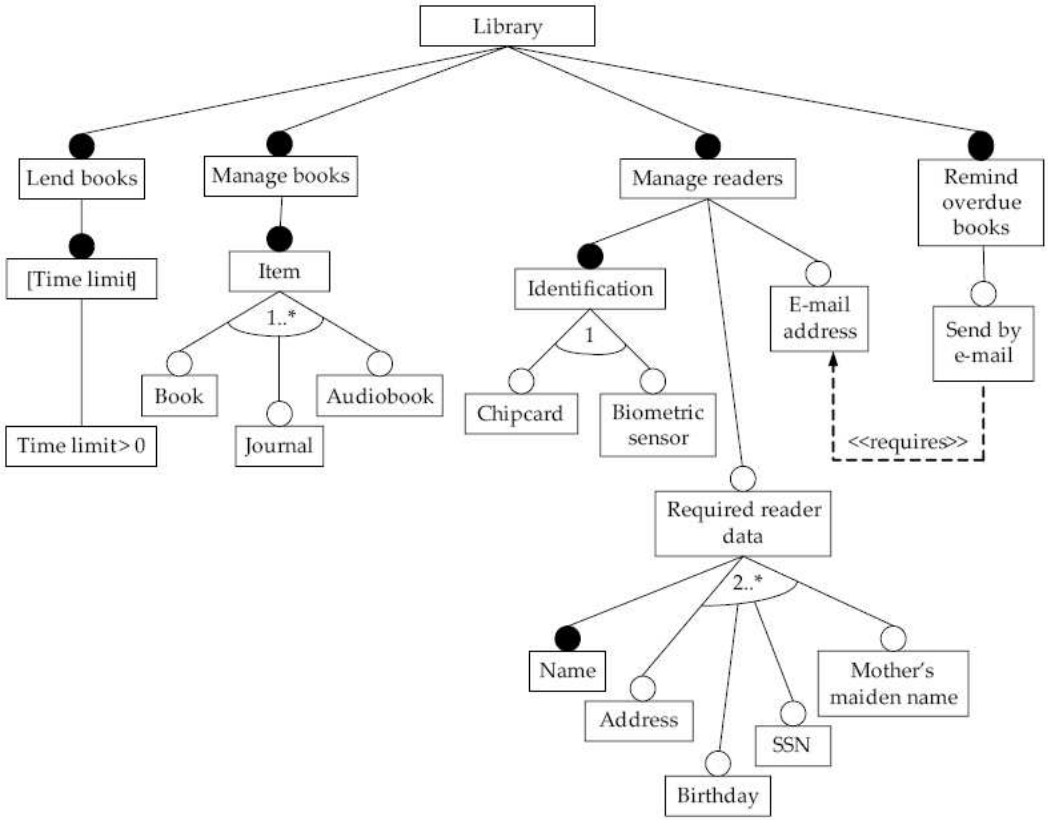

Figure 3: Riebisch-notation of feature diagram [1]

2.1.4 Supplementary Information

The feature model in order to give answers to the questions like why this feature is included, what this feature is about and when it should be selected, needs a lot of explanation or supplementary information. To include all this information with the model, separate documents are maintained [1] or attribute list is built with every feature. The attribute list too holds necessary attributes for this kind of information. There may be some supplementary information provided with features as mentioned in [10].

Semantic Description –

Each feature should have the short description of its semantics.

Rational –

Explanation of why, when / (when not) and how the feature should be selected.

Stake holders –

Every feature should hold the information of its stake holders.

Exemplar Systems –

If possible, information of the implementation of feature in the existing system should be held.

Constraints –

Information on recommendation or hints of inclusion or exclusion of feature with other feature(s) should be held.

Availability and Binding –

Availability and binding tells about when, where and to whom this feature should be available.

Priority –

Priority information can be helpful while showing the importance or necessity of the feature with software or product line.

2.1.5 General Feature Modeling Methodology

Feature analysis methodologies are mainly based on FODA. The general feature modeling analysis process consists of the following set of activities according to [1].

1. Collecting information resources
2. Identifying features
3. Abstracting and classifying features into model.
4. Defining the features.
5. Validating the features.

The information sources for developing the feature models are product documentation like user manuals, requirements specification documents, design documents, implementation documentation and source code. Other sources like text books material and domain specialists may be consulted. One should take care of the meaning of the language in the domain while finding features.

After the features are identified, now it's the time to classify them and put in hierarchy. The indication of the mandatory, optional and alternative features is done while developing the model in parallel. At the end, dependencies are resolved and any necessary supplementary information is supplied either in attribute list or in separate documentation which is properly linked to the feature.

2.2 Function-Means Tree

A function-means tree may be said as a method or technique for the representation of functional decomposition and new concept generation [23]. At the root level of tree, main functions are identified. Under each of the function, a mean (or solution element) is attached. Alternative solution elements can also be attached. Each means is then decomposed into further functions with means attached to each of them and so on up till when the decomposition is finished.

A function mean tree is basically an object oriented graphical tree representation of different parts, modules or features of a product and their relationships [13].

2.2.1 Usage History

The usage of function-means is to decompose the functionality in the form of tree to capture the complete information of design and specification in an efficient way. This usage was started by the Svensen and Hadsen [1993] where they used it for top down functional decomposition. The first major improvement in the usage and extension of feature models was contributed by [18] where they used an extended version of the function-means tree model for the design process and combined it with the `chromosome' model for product modeling. Later these function-means tree are used by the [13] in so called "computer specification model" in conjunction with the Olsson table and some modifications to the relations in order to capture complete specifications.

We will here get a deep insight into the methodology described in a paper [13] named as "computer specification model". The paper defines the criteria of this model into functions and minimum set of requirements of that completely characterize the functional needs of product design. The core model of specification capturing frame work is the function-means tree which is used for the functional decomposition of the product and the Olsson table which is used for the complete specification of requirements.

In the following sections we will explain what are the basic constructs of a function-means tree, what is Olsson table and how these two are build and used together. We will then use these methods, notations and models when implementing the case studies in function-means.

2.2.2 Basic Constructs and Development Process

Each mean in a function means tree is represented by functional requirements object FR, design solution to these FR design parameter DP and a Constraints C. Different relations are used to interlink these function-mean objects. Function-means are originally is a way to capture the knowledge with different objects.

Six different relations are used to combine different function-means or object of function-means on the different levels as defined by [13].

- a FR *is_solved_by (isb)* a DP;
- a DP *is_constrained_by (icb)* a C;
- a DP *requires_functions (rf)* FR:s on the next lower hierarchical level;
- a C *is_partly_met_by (ipmb)* DP:s on the next lower hierarchical level;
- parallel solution DP:s *interacts_with (iw)* each other;
- The fulfillment of a FR *is_influenced_by (iib)* the choice of a parallel solution (DP).

All these are shown in the figure 4. It is clear that the different function-means are organized in tree like hierarchical way. Each of these means include three objects FR, DP and C. Here FR stands for functional requirements, DP as design parameter and C as constraints. The links between these objects can be seen in the figure 4.

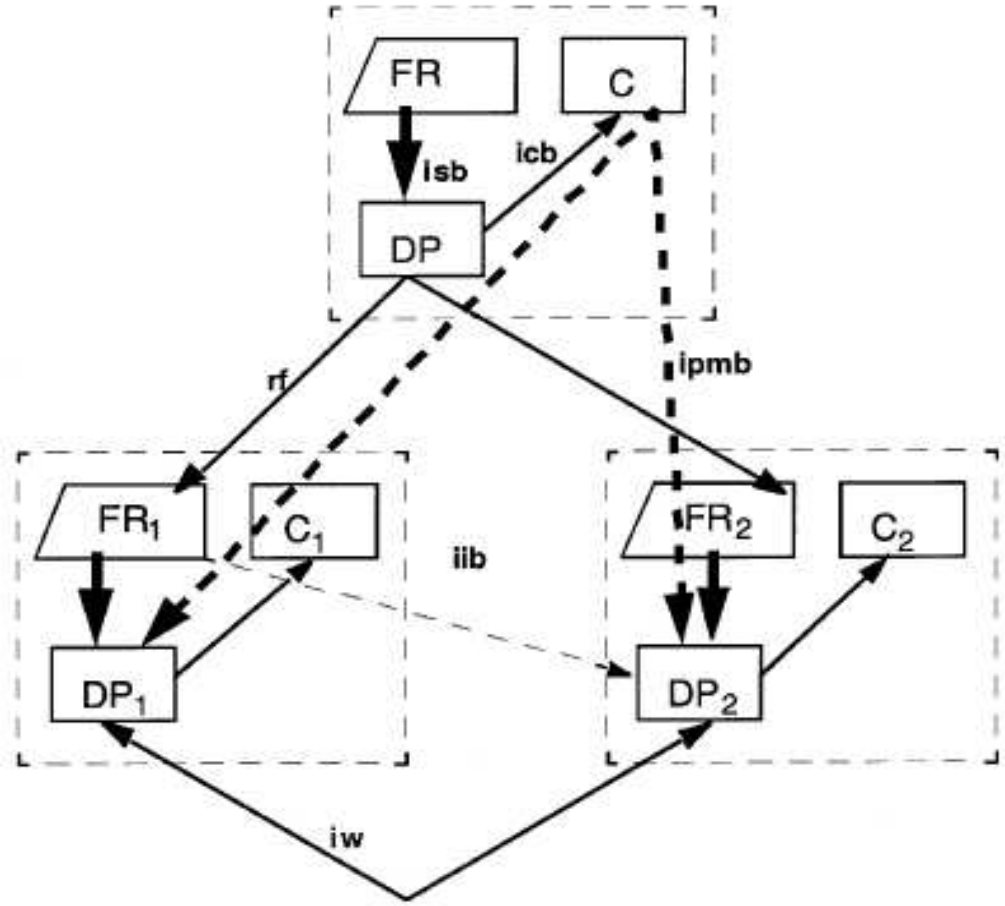

Figure 4: Basic function-means tree structure [13]

2.2.3 General Function-means development methodology

The overall process of building the function means tree starts with collecting over all functional requirements FR and then finding design solutions DP to these FR. These design parameters will most likely come up with some constraints C. Next phase starts again by defining FR_1 for C. FR_1 now requires DP_1 to be implemented and DP_1 will again come up with some new C_1 and so on.

As it was earlier said that one of the extension to modeling requirements and variability with the function-mean trees is to use the Olsson table [13] with function means to capture all possible requirements and keep track of them.

An Olsson table has the main purpose of helping the designer to put together a more complete list of criteria [13]. It consists of product life phase on one axis and important domain aspects on the other. Usage of Olsson table support the designer by asking different questions related to product under the guidance from the cells. For every mean in a function-means tree, a separate Olsson table is built and maintained and the designer when ever designing or redesigning the product looks for the relevant Olsson table, if relevant information found, it is properly linked to the related object of function-means and Olsson table is also updated accordingly.

The overall process of functional decomposition continues until the desired stage reached. At the end functional coupling is found and the objects under consideration are properly linked with *iw* relations.

2.2.4 Example Implementation of Function-means tree

In this section, we present the example used in [13] to explain the general development process of function-means tree. The authors claim in [13] that this specification model is tested with a case study involving the re-designing of the truck door. We will here use the same example to illustrate the formal flow of modeling the specification with the help of function-means tree.

The example is supported with the figure 5.

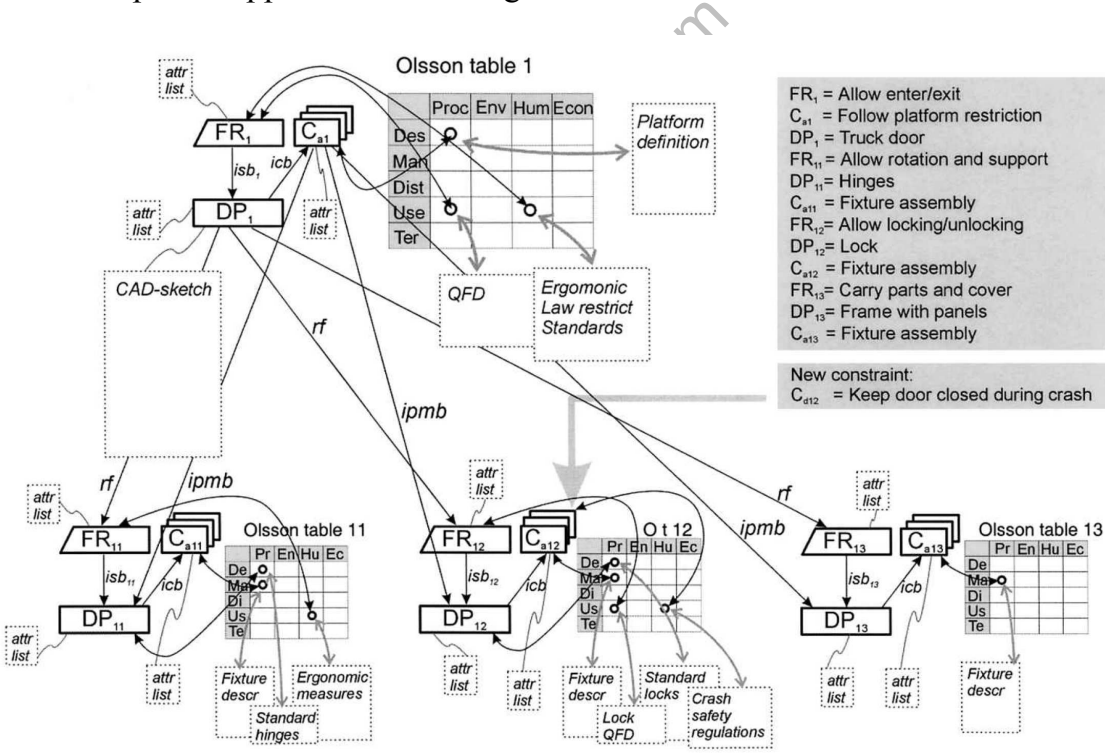

Figure 5: Function-means tree specification of a truck door [13]

It is clear from the picture that there is only a single functional requirement FR_1='Allow enter/exit' at the top level which describes the need for a door. This now needs to be further detailed and explained. Related information can be found in company catalogues and in relevant Olsson table. Olsson table 1 at this level was consulted and scanned cell by cell. New information arrived as the result of scanning the table cells, that is, while designing the door, company-specific ergonomic regulation should be considered. An attribute list is created at this point and is linked to FR_1 Links are also established between table and FR_1 object. So far, we know the constraint C_{a1}='Follow platform restrictions'.

The design parameter DP_1='Truck door' fulfilling the functional requirement FR_1 and meeting the constraint C_{a1} is now derived and modeled. This design solution may come with some documentation like CAD design. All these documents are linked to the DP_1. FR_1 is connected to DP_1 with *isb* relation, and DP_1 is connected to C_{a1} with *icb* relation.

Next sub functions are required by the design solution truck door. The sub functions are new functional requirements. These requirements are elaborated and new constraints were found. New design solutions for each of the FR and C were developed and so on until the completion of the specifications.

2.3 Configurable Components

2.3.1 Introduction

Configurable components originate from automotive car industry at SAAB in order to explore the applicability and appropriateness to adapt well-proven flow-based control methods from manufacturing industry and the application of these methods to control development of complex products [14]. Before describing the general integrated framework of configurable components, it is necessary to have understanding of some basic terms and concepts used in configurable components as follows:

According to [14], a *configurable component* is an element which provides the definition of different variants of the configuration of a system.

A system oriented view is considered to be the best as the foundation for the configurable component due to the fact that different actors are involved in development [14].

Another important concept in configurable component is *configuration of a system* which deals with the specific variant of the system and the specific variant of the system is achieved by selecting values of variant parameters. These variant parameters in turn are available in the definition of component through a *variant parameter interface (VPI)* of a component, defined as "the set of variant parameters that are available for requesting a configuration of the component" [14].

A configurable component is considered to be system construct, that is, many of the properties and features of the system should also be present in configurable component. A configurable component may contain other configurable components as elements. One restriction, that must be met, is that the actual set of components that form the composition of the parent components depends on the configuration of the parent components [14].

Configurable component modeling technique has involved many established fields from engineering design. Figure 6 shows how these established fields in engineering design provided the foundations and contexts (these are utilized and integrated with the configurable components) for the configurable components.

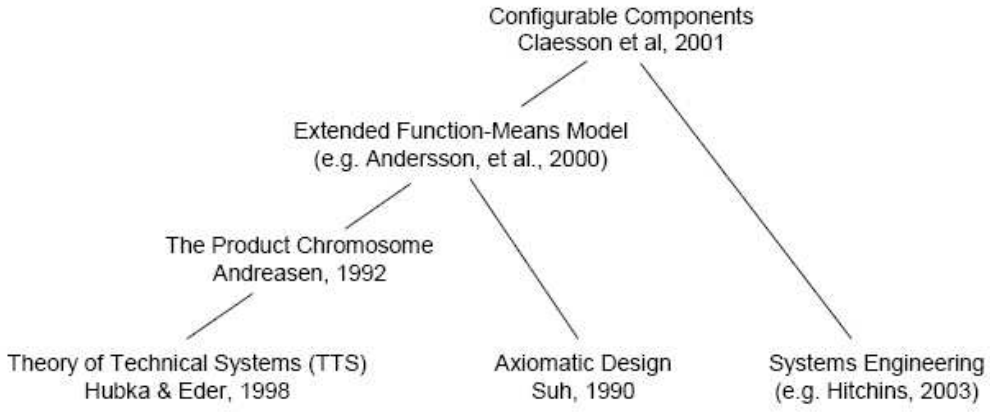

Figure 6: Engineering fields involved in CC [14]

All these fields in figure 6 require detailed knowledge which is out of scope of the thesis. The purpose to show them here is to give an overview which fields have contributed in development of configurable component. However, for simplicity, we can say that many different systems like transformation processes, human systems, technical systems, and active environment systems can be referred as Theory of Technical System (TTS). TTS are used for transformation of information (transform input into output) during design process. Chromosome product model is a generic structure and is based on theory of domain (ToD), which is used to describe a general mechanical process, and theory of domain in turn is based on theory of technical systems [14]. The description of function-means model has been presented in section 2.2.

Figure 7 shows the conceptual map of topic areas within the scope of research work presented in [14]. This framework gives an overview of the involved areas and shows how these areas are related to each other on higher level.

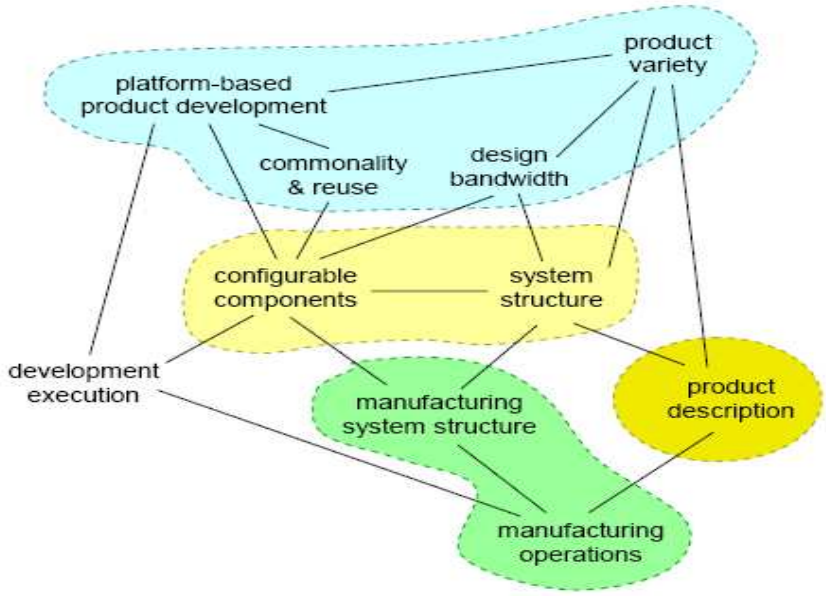

Figure 7: Concepts involved in CC [14]

Here, Product description area served as starting point of research related to configurable components. Manufacturing system structure and manufacturing operations model provided the foundations for the required integrated product and process model. The product and process models provide information about main elements of respective model and how they are connected to each other. Platform-based product development provided the facility to maximize commonality and reuse while providing product variety [14].

2.3.2 Anatomy of configurable component

Configurable components are supposed to provide a set of constructs and mechanisms. This means that configurable components will have a definition of its own elements and their functionality in terms of (i) how they relate and interact with an instance of a configurable component and (ii) how the relations and interactions with other instances of configurable components are intended [14].

According to [14], the internals (or internal structure) of a configuration component consist of the following:

Configuration rule set (CRS) provides the mapping between requested configuration of the component and the selection of a set of (sub-) components that corresponds to the request.

The configuration request is provided by the *variant parameter interface* (VPI).

The *design solution definitions* are kept as elements of its internal structure because design solution definitions are parametric designs. The configuration rule set of the configurable component uses these parametric constructs in the design solution to remove potential inconsistencies [14].

Three basic kinds of parameters can be used in the internal structure of the component as: design parameters, performance parameters, and variant parameters. Sometimes, it is very difficult to clearly define the kind of parameter e.g. a parameter can serve both as a design parameter and as a performance parameter depending upon in which context it has been used [14].

Design parameters are the characteristics of a design solution that the designer can directly influence with a decision about the value of the design parameter. Examples are functionality, interface, communication and hierarchy of concepts in software [14].

Performance parameters represent measures of purposefulness of the design solution. Examples in software industry are availability, and response time [14].

Variant parameters are constructed set of parameters (of feature) that can be used to differentiate between different designs solutions that are variations of basic design solutions. Examples may include alternative modules, limitations in control structures of software [14].

The variation is achieved using different values of the design parameters and these in turn provide varied output in the performance parameters [14].

To define sets of design parameter values, variant parameters are used.

Defining the relationship between variant and design parameters can be done either through production rules (rules from "*if*..... *then* ...."), or through function oriented approach (i.e. formulating the rule on the form "$d_i = f(vp_1, vp_2, ... vp_N)$") [14].

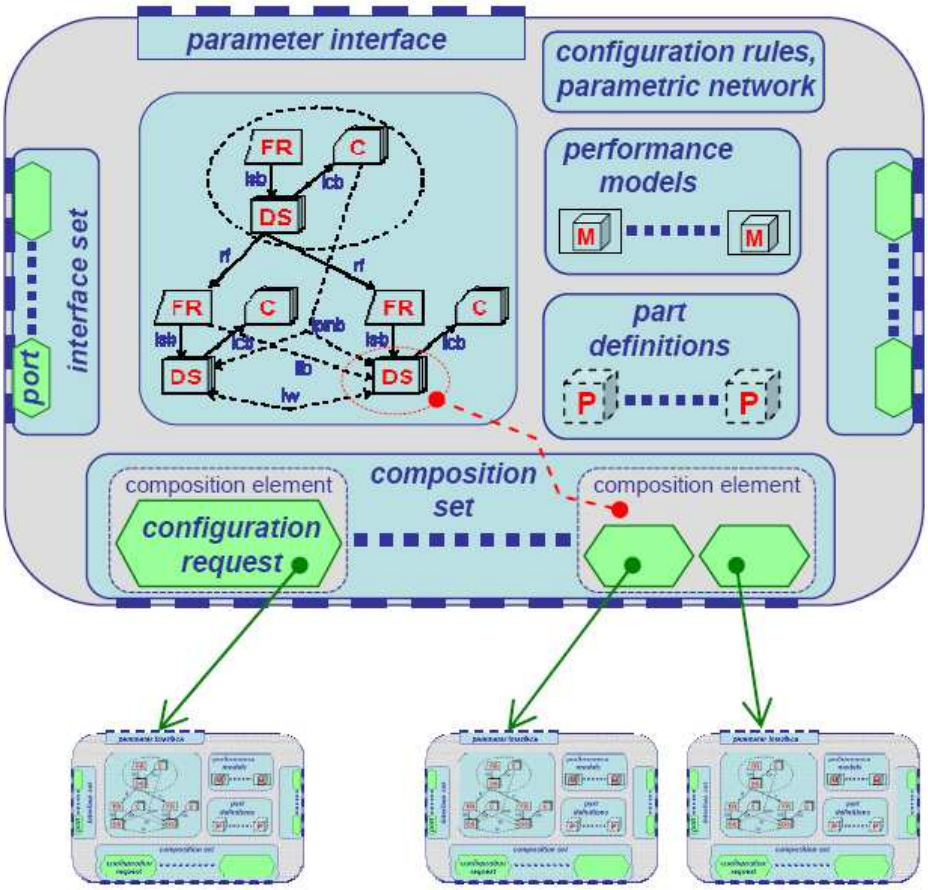

Figure 8: Integrated configurable component model [14]

The main elements of Configurable Components in figure 8 include a parameter interface; a configuration rule set; a composition set; an interface set; a performance model set; a part definition set; and a function-means model set. This function-means model set consists of functional requirements (FR), constraints (C), and design solutions (DS) and is described in detail in section 2.2.

The usage of these above elements is as follows:

The *parameter interface* is used to show parameters to its users. These parameters are defined within the configurable component. And the information about the configurable component is shown through parameter interface of the components [14].

The *configuration rule set* contains all the configuration rules or constraint network based on the parameters which are defined in the configurable components. It is responsible to provide the required constraints so that encapsulated design solutions can provide appropriate design for all variant parameter combinations. An exception is that if other components in the configurable components are themselves parametric models, then these models can also contain configuration rules and restrictions (only those that complement the configuration rule set) [14].

Composition set contains the information how its components uses other components. Moreover, it contains composition elements that establish the potential use of another configurable component. A composition element may contain the reference to alternative components. The decision whether a composition element should be included or not can be made using one of the following rules: ***Use*** <*component*> ***if*** <*condition*> or a composition element can contain references to other compositions that can be used as alternatives. But to use alternatives, composition element must also include mechanism that can choose between the alternatives. While considering alternative approach, a default choice can be selected. When a particular component has been chosen, a configuration request is issued to the referenced component [14].

Performance model is used to check the acceptance of usefulness of design method. This is done with change in design parameters which lead to different outcomes of performance parameter [14].

Part definition contains information about the part (s) for which that particular configurable component has been made. This information is stored in a database [14].

2.3.3 Applications of Configurable Components

As discussed earlier, the concept of *configurable components* was originally designed for the automotive industry. In the following paragraphs, two major approaches are presented to explain this concept which has been applied in industry i.e. a combined function-means and parametric modeling approach and configurable components for manufacturing system modeling [14].

A combined Function-means and Parametric Modeling Approach

In this approach, there are number of implementation possibilities as follows: operational implementation of the configurable components as a product definition system, a function-means based method to define configurable components, structure and parts instantiation, and platform design process [14].

In the first implementation, configurable components serve as the basics to the core product definition and description system. Figure 10 shows the operational context of the configurable components. The triangle in the figure shows the configurable product model. TAPP is a technology authorized product program

(what is technically feasible to manufacture and what are appropriate design configuration). MAPP is market authorized product program (should be sub-set of TAPP available in market as product variant). The circles represent the configurable components [14].

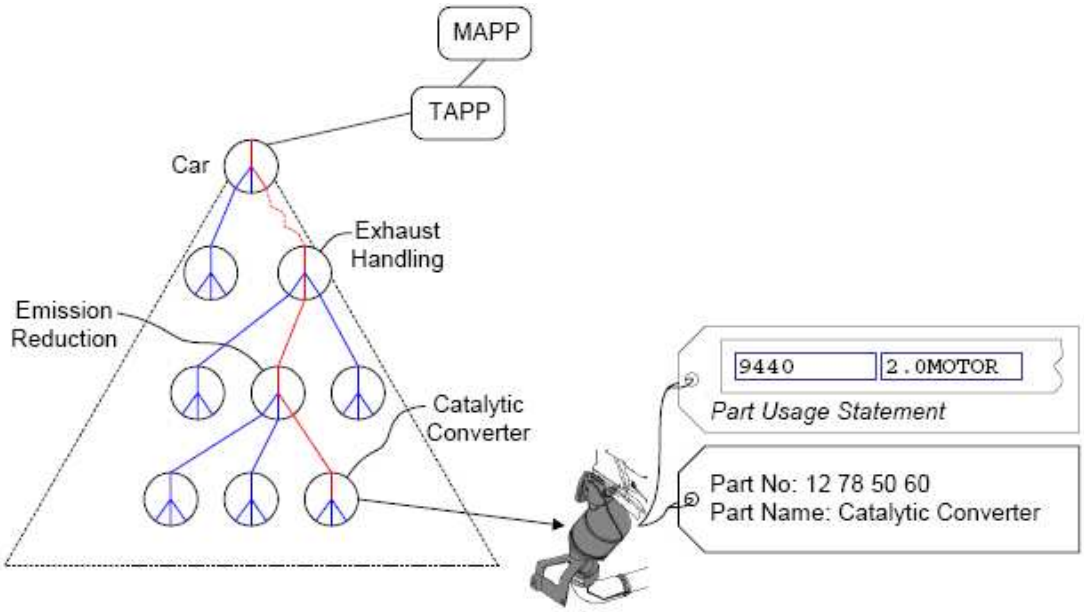

Figure 9: Implementation of CC in core product definition [14]

A function-means based method to define configurable components is shown in figure 11. Based on a market analysis based on opportunities, needs, and problems, a function-means based approach is presented. The generated *design solutions* are detailed enough to indentify individual *design parameters*. The whole process is divided into two phases. In first phase, the functional requirements and constraints are analyzed and set of design solutions is obtained and defined. In second phase, mapping of design solutions to configurable components is done which in turn is responsible for realization of physical part variants or system variant [14].

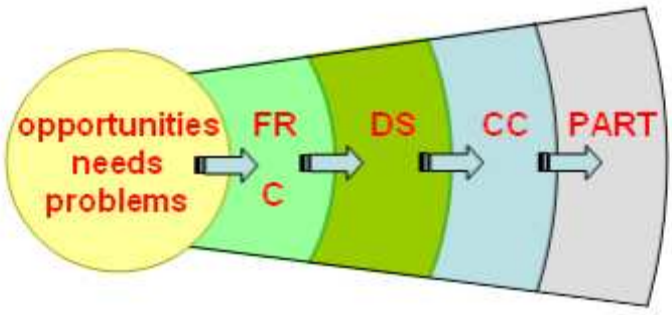

Figure 10: Function-means based method to define CC [14]

Configurable Components for Manufacturing System Modeling

This approach also includes number of implementations like understanding of *relationship between product and manufacturing system models*, *taxonomy of manufacturing operations*, and *manufacturing assembly operations*. [14]

In the first case, a usual confusion exists because of the description of different terminologies in different way. But a closer observation reveals that a harmony can be achieved by putting them in similarly structured model as shown in figure 11. The two triangles show product system structure (left) and manufacturing system structure (right) [14].

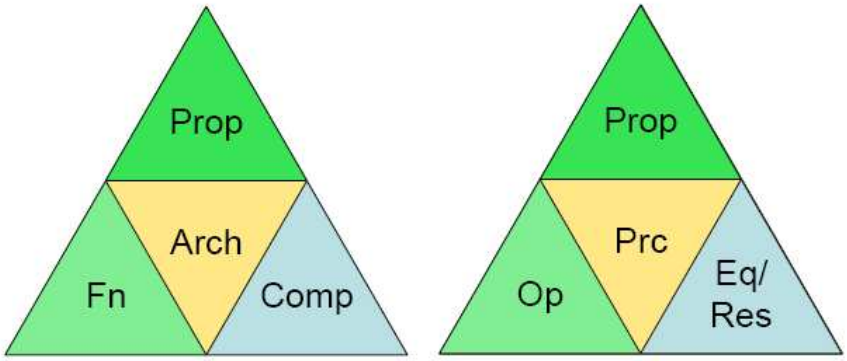

Figure 11: Mapping of product (left) and manufacturing system structure (right) [14]

According to [14], the mapping has been done as follows: the material refinement operations (Op) are mapped to functions (Fn) of a product. Organization of functions in product is obtained through its architecture (Arch) and organization of operations in a manufacturing is obtained through its process structure (Prc), hence they are mapped. Components (Comp) in a product are responsible for actualization of product and the same thing is done by equipment and resources (Eq/Res) in manufacturing system, so they are mapped too. Both systems have different kind of properties (prop) and characteristics based on the design of each system.

3 Research Method

The comparison of the available variability modeling techniques is done through the iterative and systematic review of the available literature and realising the purpose, scope, method and typical applications domain of each of the technique. A lot of relevant material supporting the decision on how to compare and how the quality of different techniques and model can be judged is available and reviewed thoroughly. The method is formal comparison on the basis of important aspects of modeling techniques according to domain, business and application area supporting by three different case studies according to domain, modeling and tool support in conjunction with the implementation of three different case studies.

3.1 Implementation

The main flow of the implementation of the method is consisting of the following set of activities:

- Decide on variability modeling techniques to study and collect the available literature regarding these techniques
- Do a comprehensive systematic review of available literature
- Decide upon a comparison framework based upon important aspects and characteristics
- Think of some case studies that can support the comparison process in conjunction with proposed comparison framework
- Implement case studies in all the variability modeling techniques
- Do the actual comparison with the help of experience gained through the implementation of case studies and the characteristics
- Propose the integration possibilities

In the next section, we will describe elements of proposed framework and their meanings in context to our research work. Following to this section, we will come up with implementation of three case studies from three different fields. After that the actual comparison is presented with discussion. At the end, there are some proposed integration possibilities.

4 Comparison Framework

During the literature review and its analysis that has been presented in theoretical background section, a tactic called comparison framework has been chosen. This framework consists of comparison of selected variability modeling techniques on the basis of the aspects and case studies presented in sections 4.1 and 4.2 respectively. On the top level, when comparing the variability modeling techniques three aspects are most important, first one is the domain of application and what they exactly needs to model. The second one is the basic constructs of the model and normal procedure which is to be followed while developing this model. And the final aspect is tools support for creating and implementing the exact model in the techniques under discussion. These general categories are then subdivided into constituent parts and described on that level. We have given these sub parts the name of characteristics. The selection of these characteristics is done through the knowledge gained in the course of literature review, application of case studies and, brain storming. The other technique which is used is that we manually compared the variability modeling techniques under discussion and selected commonalities and differences and this also helped us in defining and selecting the characteristics.

4.1 Characteristics

The detail of aspects and characteristics in order to build common understanding in this context has been described in following pages. This was done with the help of articles, and case studies given in the articles, which identify some issues regarding variability management during the product derivation [19], and some of the articles which already had done some efforts to compare and classify the variability modeling techniques [15]. This has been done during the actual implementation of proposed case studies in our comparison framework. The first important aspect of in our framework regarding variability modeling techniques is domain. Most of these characteristics are taken as inspired by the comparison criteria defined in [15].

Domain

Each variability modeling technique is mainly developed to address and capture variability in a certain type of product family and is motivated with the help of a case study in that domain in the available literature. Hence, it is obvious that one can feel much easy while applying that technique in the area rather than any other. There are some articles where it is stated and motivated which technique is best for which situation or domain, and in which case it is not suitable [1]. Domain has two major characteristics as follow:

Primary Domain of Application –

This is an important characteristic on the basis of which it can be stated which technique is best suitable for which type of primary domain or product family.

Flexibility –

It refers to how much support is available in a particular model for a particular unusual situation or domain to be modelled.

Modeling

Modeling is the most important aspect of comparison among the variability modeling techniques, this is all about how the variability is presented in a particular technique. While discussing about the modeling, the first thing that comes into mind is what steps are involved in modeling the variability and what are the choices to be modeled. Other characteristics include basic constructs, abstraction, formal constraints, quality attributes and so on.

General Development Process –

It is the first and most obvious characteristic that describes how all things are done in variability modeling technique from start to end. This is about which steps are involved or suggested in the whole process of the technique while modeling, from information gathering to classification and finally towards the complete model.

Basic Constructs –

This characteristic tell about the top level basic constructs of a model and notation which are being used in it along with other subsidiary information and related documents if they exist.

Validity Criteria –

It deals with how to validate the need and usability of the model.

Information Detail –

The level of detail a particular model holds about a particular object, product or module it presents.

Integration with External Processes –

What is the possibility to integrate a particular variability modeling methodology with a standard system development process e.g. SPICE in software engineering, or how much is this explicitly stated in the available literature of a particular technique regarding the support of an external process.

Choices –

In actual a variability modeling is making choices which facilitate extension, modification, customization, or configuration of product family artifacts in a specific context. The emergence of choices is a continuous process through all the phases of product development. In our comparison choices cover the both ways i.e. how the ability to choose is modeled, as well as, the how the possible options that can be chosen are modeled [15].

Abstraction –

Besides the complexity issues in relations and quality attributes, the issue of sheer numbers is also of considerable importance. Abstraction is used as the simplest solution. It shows only the relevant choices for the solution, example may be a layered solution [15].

Formal Constraints –

A constraint is the restriction on decisions for choices. It is necessary because not all the combination of decisions among the choice lead to consistent and useful product. The advantage is that if constraints are formalized explicitly in variability models, the consistency of configuration can be checked at hand without testing the resulting product. They help to check the consistency without checking model with software [15].

Quality Attributes –

It deals with the modeling of quality attributes in the product. Quality attributes of the product are of crucial importance and their appearance and relation with variability modeling techniques is also important to know [15].

Reasoning Style –

Which reasoning style e.g. induction, deduction, is used for a particular modeling approach.

Tools Support

The usefulness of a variability models in real life depends on its verification and measurement by the accompanying tool-suite. A tool-support deals with the creation and maintenance of variability models as well as the use of the models. Some characteristics associated with variability management can be considered as follows: [15]

Views –

Tools provide overview of variability through a user interface. The overview has one or more views where each view focuses on a specific aspect [15].

Active Specification/Inconsistency –

Inconsistency can be controlled using active specification i.e. reducing number of actions while creating or maintaining a model [15].

Configuration Guidance –

Guidance reduces the likelihood of inconsistencies or help to find these inconsistencies in the earlier stages. They can present the choices in a particular fashion which is a way to provide guidance within wide range of choices. Calculation of guidance may either be based on properties of a model or defined statically as a strategy [15].

Inference –

Inferences (decisions on the basis of constraints and/or previous decisions) can be made with the help of inference engine [15].

Effectuation –

Effectuation is mapping of decisions into actual product family artifacts [15].

4.2 Case Studies

Second part of our comparison framework consists of case studies from three different domain areas. These case studies will help to describe and understand the underlying nature of domain areas and appropriate solution using above mentioned variability modeling techniques mentioned in section 2. The first case study is from automotive industry where the variability modeling of car door has been managed. Second one is related to software product lines where the variability of an email client has been exemplified. The third case study covers the area of service oriented system where variability of library services has been modelled using all above three techniques.

In the following sub –sections, the detailed description and implementation of the case studies has been presented. We have not used any of the modeling tools while implementing these techniques but only have applied paper-based modeling. However, these case studies have been designed and explained in such way that they fulfil all the clarification of the relevant situations and are implemented utilizing all important aspects and notations of each technique. Due to the shortage of paper size not all the variability is modelled in a particular case study but only important features have been shown.

4.2.1 Platform-based Product Development - Automotive Industry

Main problem regarding product variability in automotive industry is how can high variety, modular decomposition and platform-based products be represented during product development that provides support through-out the entire product life cycle? As described in purpose that product variability is a serious challenge to the developers of complex systems (for example automotive industry) offering rich set of features, which make it difficult for their developers to achieve flexibility and restrict complexity at the same time. So, it is necessary to find solution for it. The first simplified application case deals with variability in a car system. There are so many options for this variability and also from different perspectives. For example, door system, electric system, Engines and exhaust system etc. Every system has different parts and each part is available from different manufacturers with different configurations. Another aspect is that different car models require different parts with some variations from other similar part (also when a new model is introduced). Although there are many systems that can be considered but for simplicity we will took one system i.e. design of car door system.

This problem arises while designing structure of a car which has one definite aspect i.e. design of door system. The system is related to manage variability of car door system i.e. to manage variability of different parts in a door system like locks, frames, panels, keys, windows, and handles. There are many things involved while managing variability of each of the part in door system i.e. area, mass, size, width, thickness etc. of each part. These subparts or subcomponents may become to hundreds or thousands in quantity. We will remain focused on the top level general aspects of car door design in the following implementations of card door in different variability modeling techniques.

Car Door –Feature Modeling Implementation

The example implementation of feature modeling of the case study Car Door is depicted in the following figure. It is clear that Car door have the features like Open/Close, lock, frame with panels and window. These features are mandatory and hence, shown with the filled circle. Open/Close have other two sub features rotate and hinges which are again connected in a hierarchical structure with the filled circles for being mandatory features. Lock has the mandatory sub features child protector, key and handle. Frame with panels again have the mandatory sub feature which allow the window to open and close. The last feature Window is again classified with the two alternative sub features manual and power. These features are alternative and are shown with the multiplicity 1, which means than only one feature at a time is to be selected in order to be applied on the window. It is obvious from figure 12 and description of features above that this case study focuses the car door from basic structural point of view.

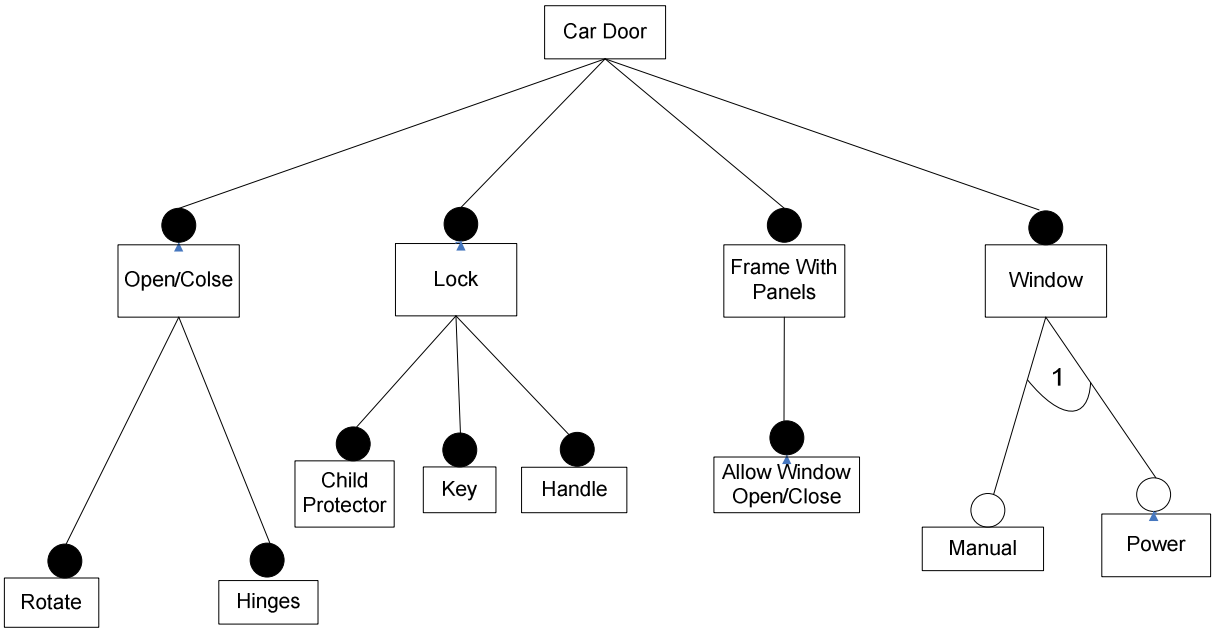

Figure 12: Feature model implementation of car door

Car Door –Function-Means Implementation

The function-means implementation of the case study of the car door is partly adapted from the example case study given in [13]. The reason is that this case is implemented at large scale and has been tested. We will have some already tested and analysed summary and results from the paper too, which were concluded by the authors.

At the top level, functional requirement of the car door from structural and innovation redesign point of view is FR_1='Allow enter/exit'. As we can see, this requirement is modelled as an object in the function-means. Now, Olsson table belonging to the first hierarchical structure is scanned through. In one cell there is reference to some company-specific ergonomics regulations to be considered, and in other one there is reference to QFD (Quality function deployment "a method to transform user demands into design quality") analysis results for the car door as guidance. These documents are to be consulted and then an attribute list to be created and linked to FR_1. Olsson table cells are also linked with FR_1 object and updated with a reference to the new available material if it was previously unavailable.

Now after reading all this material, the constraint C_1='Follow platform restrictions' is found. To acquire more knowledge Olsson table is again scanned and relevant referenced documents are consulted and their corresponding cells are linked to C_{11}.

The solution to the functional requirement FR_1 and meeting the constraint C_1 at this point is derived as design parameter DP_1='Truck door'. This design parameter object is further described with the attribute list and a computer-aided design (CAD) of the door. This attribute list and design sketch is linked to DP_1 object in function means and so far modelled objects are connected to each other. FR_1 and DP_1 are connected with *isb* relation while DP_1 and C_1 with *icb* relation.

Next step is to think about the sub-requirements to model and implement the DP_1. This is done by the company experience, intuition and with help Olsson table. Here, we are going to present only three out of many formulated sub-functional requirements. In figure it is shown with FR_{11}='Allow rotation and support', FR_{12}='Allow locking/unlocking' and FR_{13}='Carry parts and cover'.

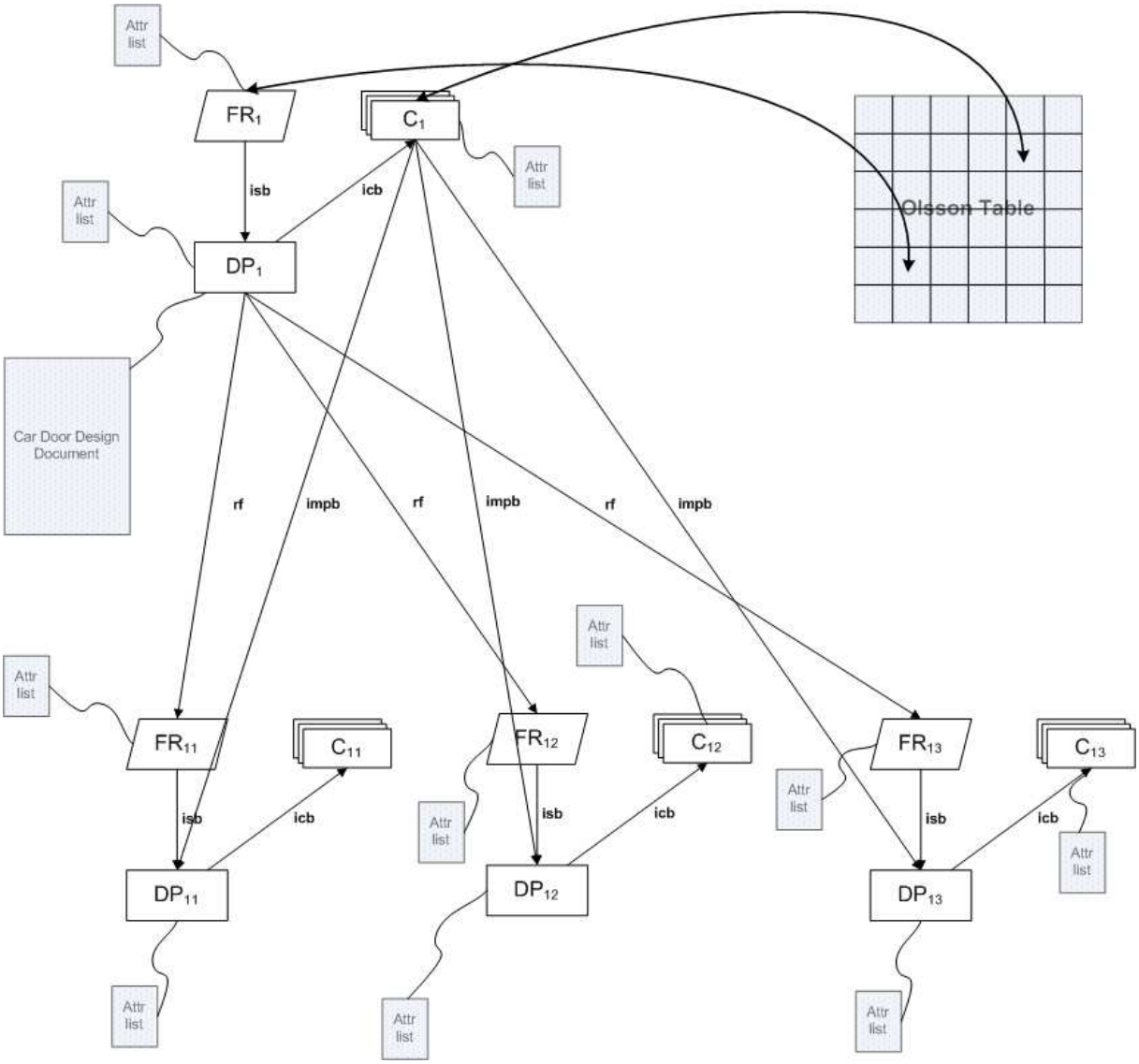

Figure 13: Function-means implementation for car door

These all sub-functional requirements link DP_1 with *rf relation.* These FR's are constrained by C_{11}='Fixture assembly', C_{12}='Fixture assembly' and C_{13}='Fixture assembly' which are decomposed consequences of the constraint C_1on the previous level. Furthermore, at this stage again the Olsson tables are built for these function-means or consulted if available. Relevant cells are scanned and if new constraints are identified, they are linked.

Now at this point when after sub-functions and sub-constraints are identified, sub solutions DP_{11}='Hinges', DP_{12}='Lock' and DP_{13}='Frame with panels' are derived. These solutions are supposed to fulfil the corresponding functional requirements at this level. Relations between different object are established as done in previous level, only one additional relation *ipmb* is established between C_1 and all of C_{11}, C_{12} and C_{13}.

Car Door – Configurable Components

Solution to Car Door using configurable components is supported by the figure 15 and explanation below as:

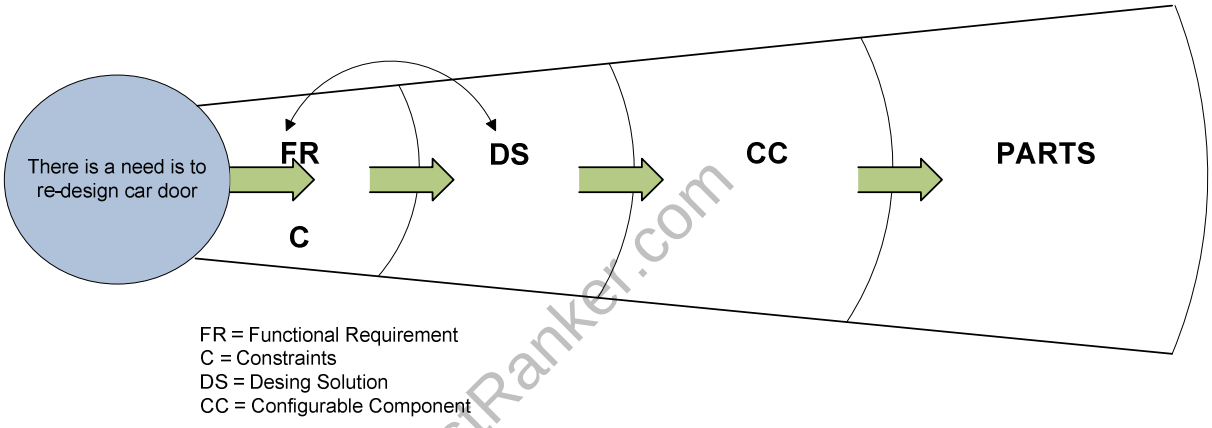

Figure 14: CC implementation for car door

Figure 14 describes the solution at higher level. Based on the analysis, there is a need to re-design the car door as mentioned in the case study. The function-means approach (for detailed description of function-means, see implementation of the case study using function-means approach) is applied such that the design solution is detailed enough to be analyzed from the standpoint of variety. The design solution (DS) used here is parameterized DS.

The solution is provided in phases as follows:

In the first phase, the functional requirements and constraints are thoroughly analyzed and set of design solutions is created after analysis. The variety in terms of functional requirement is considered to provide the variability in overall modeling. In the second phase, design solution is mapped to configurable components. Every configurable component implements the design solution. The configurable component also includes sufficient sets of design solution so that configuration provides enough information for parts realization.

An important thing to know, before going into the details of model presented in figure 15, is that each configurable component (CC4 and CC5) has all elements as described in figure 9. The detailed model is presented in figure 15 where configurable component (CC4) is an implementation of sub-design solutions DP_{12} and DP_{13} (lock and frame with panels respectively) while configurable component (CC5) is an implementation of sub-design DP_{11} (fringes).

To implement DP_{11}, CC5 include parameter interface, configuration rule set, part definition and composition set. Parameter interface is used to show the value of basic parameters (design, performance and variant). The basic parameters are used to describe the design (thickness, length, area, mass etc.) of fringes by providing enough information required for the design of fringes. Composition element of CC4 in this case is not requesting for inclusion of any other configurable component. But composition element of CC2 is requesting for CC4 and CC5. Configuration rule set include all the constraints related to 'fixture assembly'.

CC4 is a set of two configurable components and provides implementation for lock and frame with panels.
The physical part variant is physically realized by a certain configuration of the system CC.

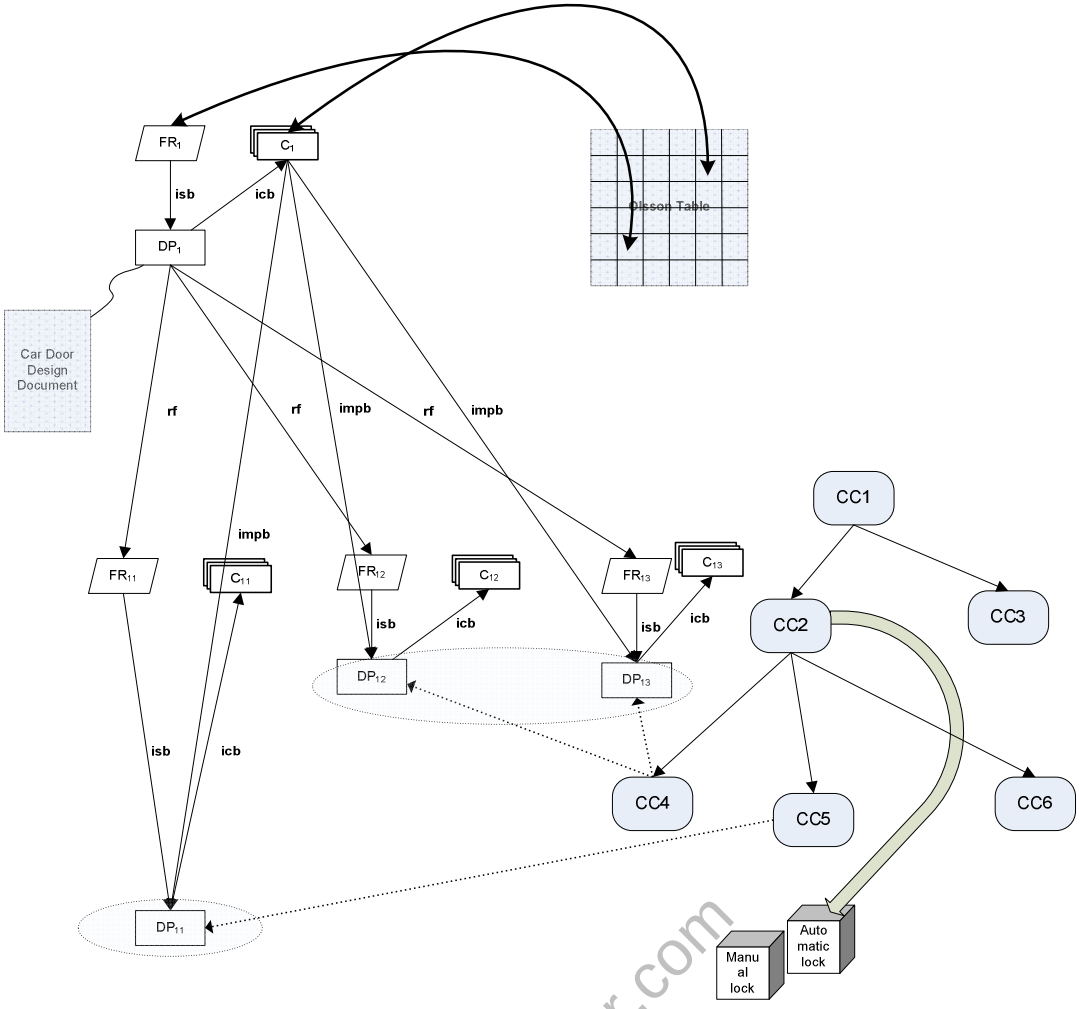

Figure 15: CC implementation of car door

4.2.2 Variability Modeling in Software Product Lines

Software variability management is the key to the reuse of software components. We will here present the case study of an email client which is used to receive or send electronic mails. There is a variation in between different components of an email client like which number of services, editor type, connection type and the operating systems, it supports. It also has the variation in graphical user interface, type of end user usage policies and type of information it saves or displays. Variation also includes the area of customization according to the user preferences and adaptability to the hardware where it is going to be accessed. There may be variability in connection types and communication protocols as well.

So all this makes the email client a strong candidate to be included as a case study for the application of different variability modeling techniques and will be hopefully enough to present the software product lines area in our case studies.

For the sake of simplicity here, we will use the more general and top level system components of email client and will use feature modeling, function-means tree and configurable components to model and visualize the variability.

At the very top level the mandatory features of an email client includes the connection, message editor, message sender, message receiver and operating system compatibility. A connection may be LAN, wireless LAN or GPRS. A message receiver works with two types of protocols IMAP and POP3. Operating systems may include Windows, Linux and Symbian.

Following is the implementation of the email client in the above mentioned three major techniques:

Email Client – Feature Modeling

At the top level an email client is shown in the figure 16 as a feature. The next sub-features connection, message editor, message sender and so on are mandatory features and so these all are shown with the filled circle. Email client must support at least one or all of the operating system. Windows, Linux or Symbian are the operating systems in the figure. These are alternative features and so are shown with the arc, multiplicity in the arc is shown as 1..*, which means that at least one or more than one operating systems are to be supported in the email client. Message receiver requires the implementation of IMAP protocol as mandatory feature while POP3 as optional feature. POP3 protocols requires external desktop application feature which is shown with the arrow going from it to external message editor in figure 16.

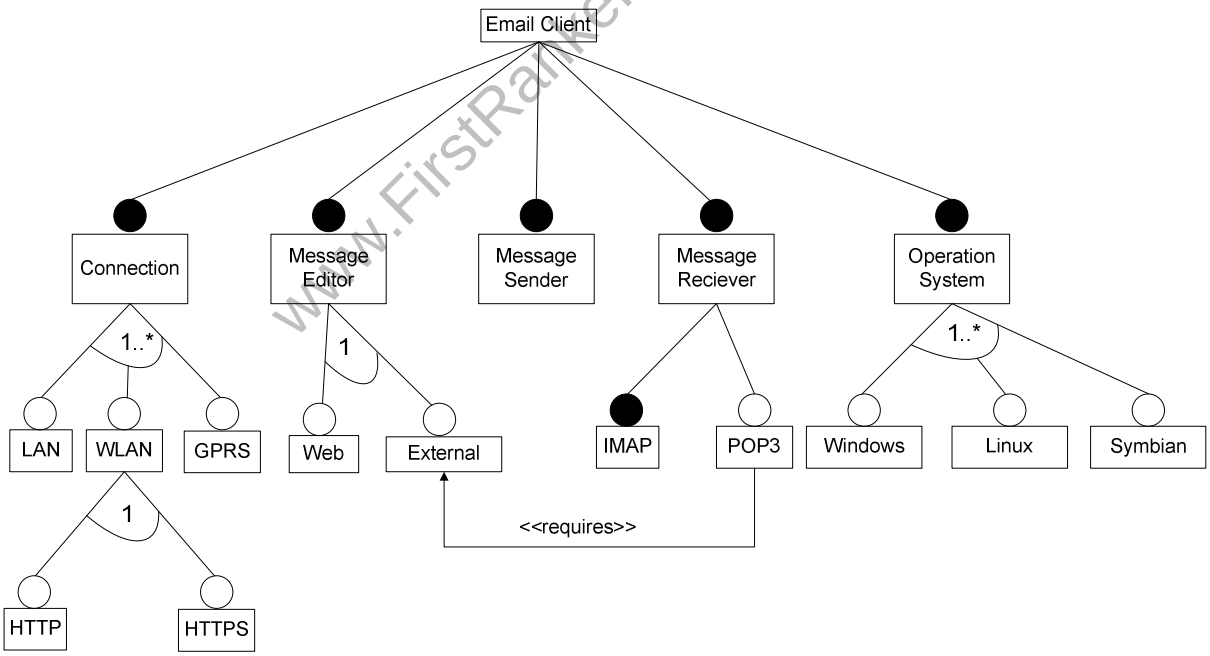

Figure 16: Feature model implementation of email client

A connection may have the LAN, WLAN or GPRS connectivity with the server as alternative features. Again WLAN requires the implementation of HTTPS or HTTP protocols as alternative features. Multiplicity here shows only digit 1, which means only one out of all these will be implemented at a time.

Email Client –Function Means Implementation

The function means tree implementation of the email client starts with specifying the top level functional requirements FR_1 = 'Email sender receiver needed'. Functional requirement is modelled as object. It now needs to be further described and decomposed. For this, the required standards and needs of necessary components for building an email client should be read at this point. If an Olsson table is available it should be consulted [13], all the required references to the documents may be found in this table, otherwise one should start thinking how to build the email client keeping in mind the earlier read documentation and requirements. If something was found in table, then put the reference of the cell into the attribute list. We suppose here that email client was previously built and we have the necessary documentation and description regarding needs, standards and others required material along with the pre built Olsson table. So we will follow the approach of redesigning of the email client.

After going through the whole process previously described, we came with a software requirement specification document at general level. Olsson table was updated by putting the references of this document in appropriate cells; table was also linked to the functional requirement object FR_1. Alongside we came to know that there are some constraints which must be solved. Constraint C_1='Software which can communicate over network is needed', with some other details was known at this point. To acquire more information about this again the Olsson table should was scanned and updated and with the found documentation. A link was also established between C_1 and the table.

At this point, we came across a design solution to above FR_1 with a design parameter DP_1='Email client software' which fulfils the functional requirement at this level. At this level a software specification document was built. This document was linked with the attribute list of the DP_1 and the Olsson table was updated accordingly.

Until now we have a FR_1 which is constrained by (*icb)* C_1 and solved by *isb* a DP_1. This is shown in the figure 17.

Now sub-functions of 'Email client software' must be defined. This is done by the designer's experience, available scripts of end user feedback and documentation read in the previous whole process. Now again on the next level we came with the functional requirements FR_{11}='Make connection to the server', FR_{12}='Provide message editor' , FR_{13}='Provide message receiver' and FR_{14}='Provide support for operating system' which are constrained by C_{11}='Protocols to be implemented', C_{12}='Editor should support multi languages', C_{13}='Support IMAP protocol' and C_{14}='Should be portable' respectively. This all is done by again following the whole procedure of looking for Olsson table for every function-means. If available consulting it, otherwise looking for relevant material in other sources and then building and updating the table and making links between table and function means object. After doing this whole process we came up with the design parameters DP_{11}='Connection Module', DP_{12}='Message editor module', Dp_{13}='Message receiver module', and DP_{14}='Different modules for every type of OS' along with some design documents, restriction and standards. All this material was properly linked and referenced in Olsson table and in the attribute list of function-means objects.

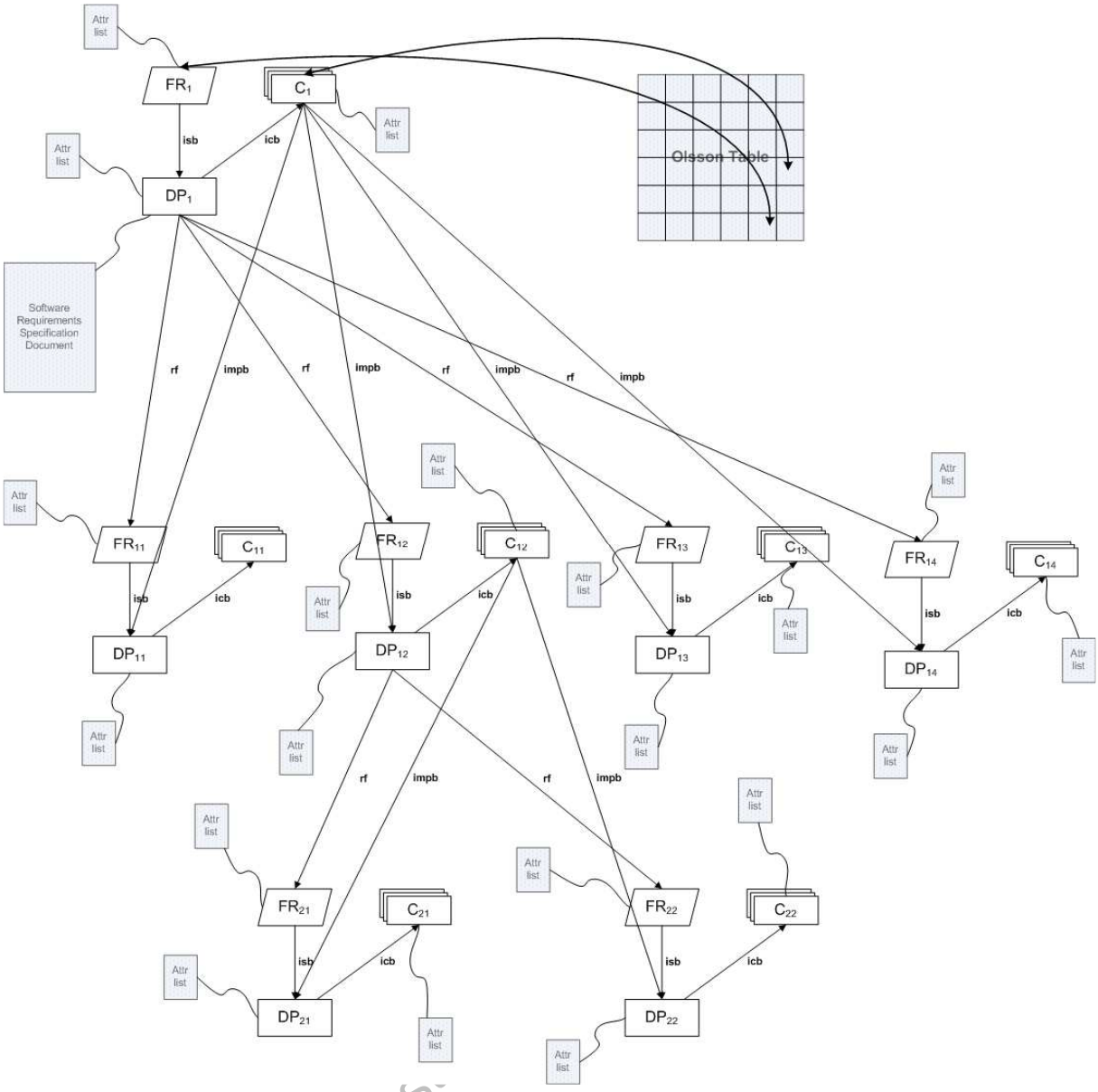

Figure 17: Function-means implementation of email client

As we can see from the figure 17 that DP_1 is connected by the link *requires-functions(rf)* with FR_{11}, FR_{12},FR_{13} and FR_{14} and constraint C_1*is-partly-met-by(ipmb)* the design parameters DP_{11}, DP_{12}, DP_{13} and DP_{14}.

Now if we look at DP_{12}='Message editor module' it *requires-functions (rf)* FR_{21}='Browser application needed ', and FR_{22}='Stand alone application needed' which are *is_constrained_by (icb)* C_{21}=' html support' and C_{22}='Desktop application' respectively, and *is_solved_by(isb)* DP_{21}='Editor web page', DP_{22}='Editor software application'.

These design solutions will be further decomposed if required and will come with the new functional requirements and constraints which are then met by design parameters and so-on. But the scope of our case study ends at this level.

At this level, functional coupling is checked and two objects of design parameter if required are linked with each other by the relation *interacts_with (iw)*.

Email Client – Configurable Component Implementation

Configurable component has its roots in manufacturing industry especially automotive car industry as described in the theoretical background above. This modeling technique is relatively new and has not been implemented in other manufacturing industries as well as in software industry. The concepts behind the solution to this case study include knowledge from the configurable component thesis, XML-based Variant Configuration Language (XVCL) and other contributions from the software engineering [21] community regarding product lines of software products as suggested by the author in [14]. The system structure in configurable component is similar to x-frame (Meta frame) in XVCL. A detailed description of XVCL and software product lines is available at [20] and [24]. Although Software product lines take into consideration the business, technical and customer aspects, but here we have considered only the technical aspect for simplicity.

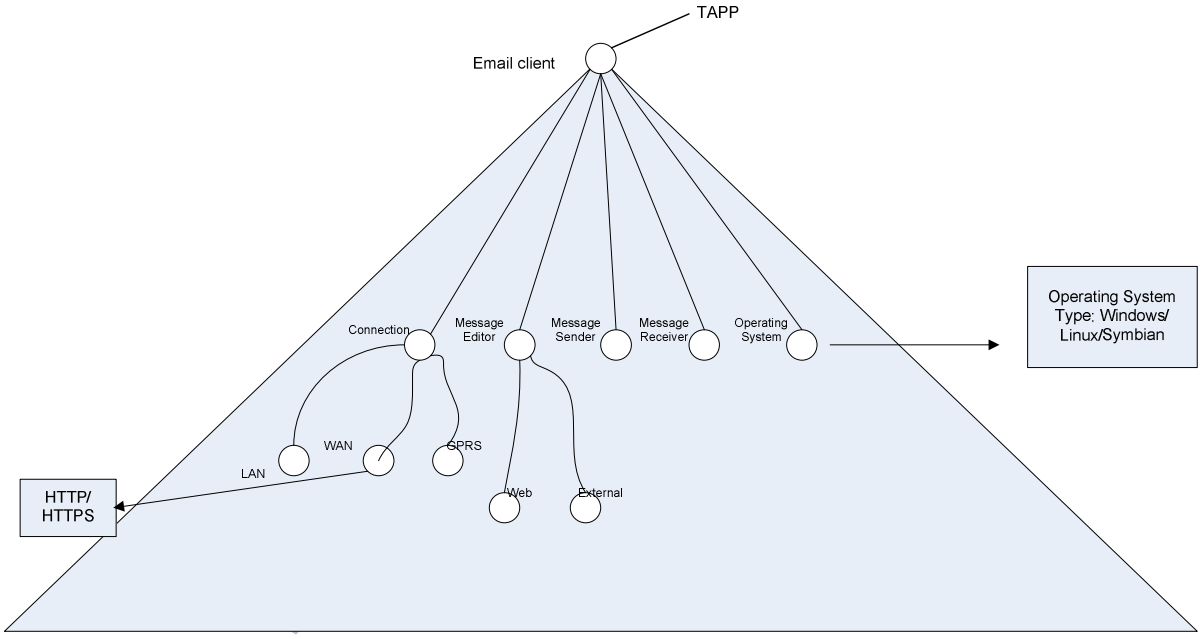

Figure 18: CC implementation of email client

The circles in the figure represent the configurable components. TAPP is technically authorized product program i.e. technically feasible to manufacture and validated as appropriate design configuration. Every configurable component has reference to other configurable component in order to take decision whether to include or not. The approach used to include reference is to have a default choice. The decision is usually made on the basis of component-if-condition.

4.2.3 Variability Modeling in Service Oriented Systems – The Library Services

Variability modeling of the services provided by organizations may be very helpful while satisfying and analysing the customer needs, wished or demands. It may also be helpful while systemising and reproducing the large scale service oriented systems if proper variability of services that are being, should or already provided is modelled. Thus variability modeling of service provided by a service oriented system is as important as variability modeling of product lines of a family of products.

In this case study we will illustrate the variability modeling of services provided by a library in three different variability modeling techniques.

A library is basically an organization which has its own assets like buildings, staff, books, journals, and/or electronics products. A library also has its members or customers and the library provides different types of services to its members. These services may include lending of study material, intra library loans, search facilities for papers, books and journals in different data base and library catalogue, printing/photocopy facilities and internet surfing facilities. Also there are other types of services like providing study places and providing special types of equipment which facilitate special people while studying.

Thus the library usually include a wide range of services offering to customers, but we will remain specific according to the services we have defined in the introduction of this case study for the sake of simplicity and limited because this model grows so rapidly that a paper do not comply with it.

The Library Services –Feature Modeling Implementation

Feature modeling of library services is purely from the service oriented point of view. As according to the figure 19, we can see that the library services include study material, printing, internet services and booking of study places as mandatory features and intra library loans as optional features. Intra-library loans are those study materials which a specific library do not have itself but a customer can search the other partner libraries catalogue and can request for that material, library then will manage to borrow that and lend it to the customer. As we can see this feature is not free.

Study material is then can be printed, software or audio/video. Printed material is mandatory but software and audio/video are optional and alternative features. A printed material may be issuable or non-issuable. Software includes CD's and downloadable as alternative features.

Printing has then conditional features to have credit in accounts which again have a parameterised feature which requires the credit above than zero.

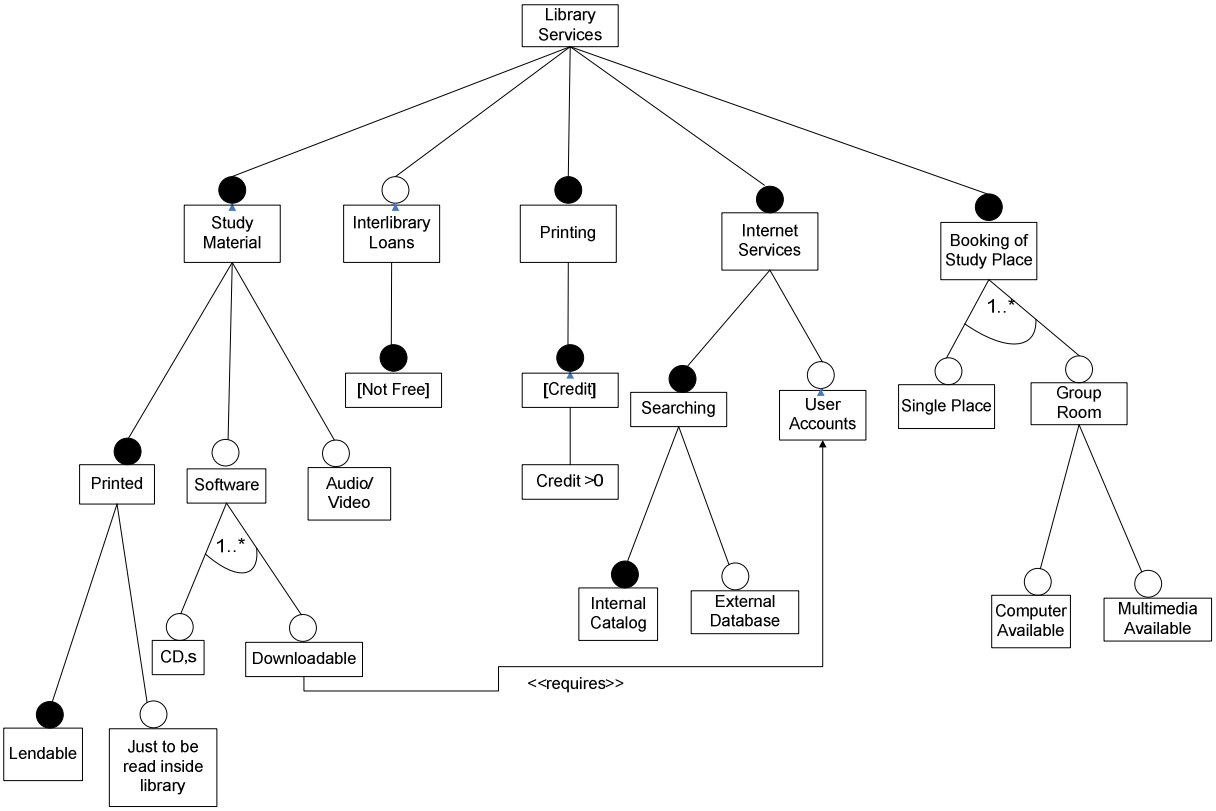

Figure 19: Feature model implementation of library services

The Library Services –Function Means Implementation

Starting point for modeling the specifications of the library services is functional requirement FR_1='Establish Library'. How to establish a library? ! O.K., the answer to this question comes with the things in mind like having a building, having staff, having books and having customer and studying the related material and consulting the Olsson table, but there is something which is to be defined that which services, we need to provide to customers. At this point we come with constraint C_1='Define and implement services'. We now think on it and decide to define the services and implement them all, at this point we will have a document called 'Library Services' which will have the information about basic services provided by the library. This is the design parameter DP_1='Library Services'. At the same time Olsson table is updated and the links between document and table and function means objects are established.

This design parameter DP_1 require functions FR_{11}='Provide Material', FR_{12}='Some users want prints' and FR_{13}='Academic oriented study needed' which are again constrained by C_{11}='Some customers do not have time to come to library', C_{12}='Printing costs a lot' and C_{13}='Study places are not sufficient' respectively.

These functional requirements are then solved by design parameters DP_{11}='Provides issuable study material', DP_{12}='Printing facilities are not free' and DP_{13}='Provide booking mechanism of study places'.

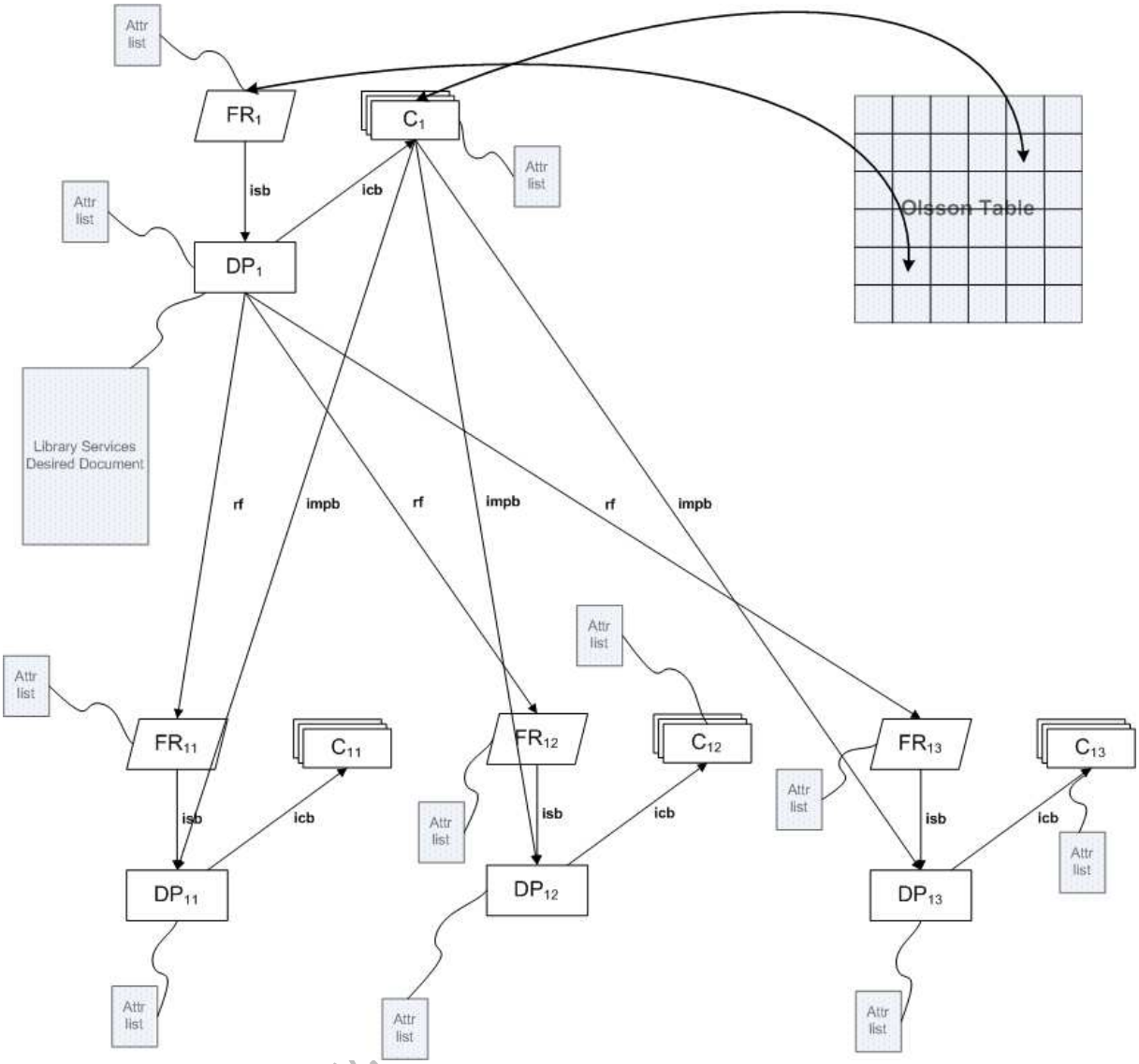

Figure 20: Function-means implementation of library services

The Library Services – Configurable Component Implementation

There is no specific solution to library services using configurable components. The reason for this is that no guideline or references are available for this type of solution. The only possibility for the solution is one similar to email client system. The purposed solution in figure 21 is the result of brainstorming on the basis of literature review and practical examples of configuration component and of-course email client. Here again the circles represents the configurable components. The restrictions or conditions like for printing services, printing credit should be greater than or equal to number of prints to be taken, are defined in configuration rules. Another restriction for interlibrary loan, i.e. interlibrary loan is not free, is added to configuration rule of interlibrary loan configurable component. The restriction for usage of internet service is that a user must have a valid user account and password which is again added in relevant configurable component.

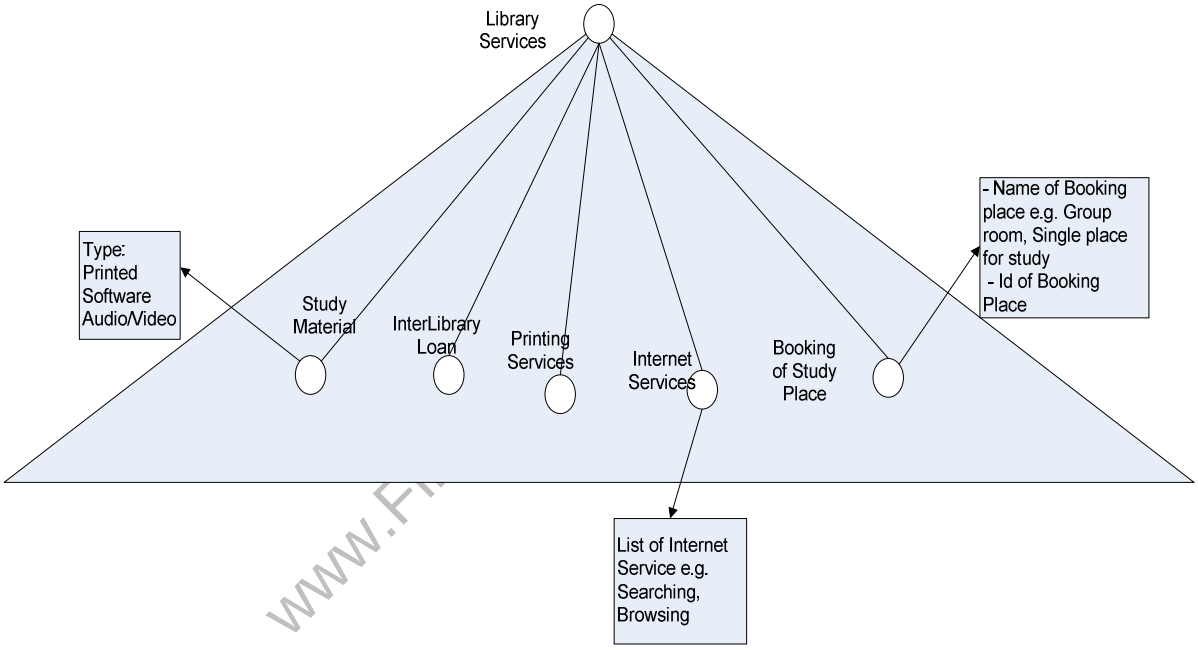

Figure 21: CC implementation of library services

5 Results

The overall modeling and application of the three different case studies in each of the variability modeling techniques and the previous literature review has resulted into the fact that these techniques have commonalities and differences among them. But what these commonalities and differences exactly are and how does that make them suitable for different types of product families or situations. To answer this question, we will use our comparison frame work and will apply the characteristics defined earlier to see the commonalities and the possible ways to integrate these techniques.

So far, a common understanding of attributes has been established and modeling techniques have been explained with the help of cases studies. An important question in this context is which technique is the best suitable for which situation and is there any integration possibility of these modeling techniques. This chapter sheds light to answer the question in two steps: first, the comparison (similarities and differences) is presented among the above mentioned variability modeling techniques on the basis of characteristics and case studies presented to find out the best modeling technique. Second, this comparison result is used along with literature review and brain storming to find out the integration possibilities of these variability modeling techniques.

The main aspects on which comparison is based are domain, modeling and tool support.

Domain

Most of the variability modeling techniques were developed to solve a particular type of problem domain, and in literature are exemplified with the help of small case studies. Although, case studies chosen and implemented in this paper are enough to give a bird eye-view of the techniques under discussion, but they do not put insight into the large scale implementation of that particular area. But, after the implementation of case studies it can be said that which of these techniques was best suitable for which domain while managing variability in the fields described in case studies and which type of information or relations these capture.

Although, in some literature the best suitability [1] of a particular variability modeling technique for a particular domain area is suggested, but it depends upon the properties of the product family and the situation at hand.

Primary Domain of Application –

From the case studies elaborated in this thesis, there is a clear indication that integrated model of configurable components is best suitable for platform based product development where it was originated.

On the other hand, for software product line (email client) function-means tree was the most suitable because it captures all the variability in requirements constraints and design in an efficient way, where as feature models only capture the top-level functional variability from a particular customer/engineer requirement point of view. Configurable components require lot of work to be done for the implementation of this particular domain application using contributions from software engineering with common methods from electrical and mechanical engineering as suggest by author in [14]. But an attempt has been made to implement such case study using software product line from software engineering discipline and product definition concept from configurable components. This implementation attempt was quit a beneficial in capturing the specifications along with structure variability.

Feature models however are more looking to capture product structure along with the variability.

Flexibility –

It is also very important for a variability modeling technique to comply with changing situations and flexible enough to adopt new things. Both of the feature modeling and function-means tree are to some extent flexible to changes within the domain area. For example, in the case study of an email client, if the new requirements due to technology change arrive, there is a clear tendency of accommodating these changes in both function-means and feature model. Also implementation of all of the case studies was not a big hurdle however every technique captures different type of information from case study.

Configurable component provides flexibility to greater extent than other two models with the use of parametric constructs in design solution that remove potential inconsistencies [14]. Moreover, configurable components use QLE (quality, lead-time, and effective/efficient) model [14] in terms of market behaviour and customer preferences that influence the variability. This model ensures (i) the stability of products quality when market behaviour and customer preferences are changing, (ii) robustness of system response to changing demands, and (iii) response effectively and efficiently to changing business environment.

Modeling

The actual modeling of all three described variability modeling techniques vary from each other in terms of general development process, basic constructs , validity criteria, and information detail. However, there are some commonalities in abstraction, formal constraints and the hierarchical nature of the models.

General Development Process –

The general development process of these techniques differs from each other. In feature modeling the main process include literature review, identification, classification, modeling and definition of different features and building it in the form of tree with the proper notations, handling multiplicity and exclusiveness. While in function-means the main process emphasize more on capturing the complete requirement specifications and involves gathering of top level functional requirements, fetching relevant constraints and producing corresponding design solutions, again sub functional requirements on the next level and so on. On the other hand, configurable component modeling technique introduces the product variants earlier into the development process and to a higher level of abstraction with the help of configurable components as the classification is done in feature models. These components may then choose other components. They provide set of constructs and mechanisms. A configurable component encapsulates design solution definitions as elements of its internal structure which is similar to feature definition in feature models.

Basic Constructs –

Feature modeling – Feature tree, with nodes as entities and edges as relations, different notations used for representing the relations like mandatory, optional, and alternative features.

Function-means tree – Tree with function-means, each of mean consisted of three objects which are again interlinked with the object within same function means or in some other function-means.

Configurable components (CC) are the basic constructs/elements for configurable component modeling technique. Each CC includes function-means tree (optional), configuration rule set, parametric network, parameter interface, part definition, and composition set.

The basic constructs in feature modeling and function-means differ from each other with respect to visibility and information they intend to capture. Although there are implicit similarities like attribute list and links between features or function-means objects. On the other hand, configurable components include constructs of function-means as optional part and also static structures of configurable component sounds like a feature model. These static structures are part of variant bill-of-material as described in upcoming section 5.3.1.

Validity Criteria –

Unfortunately there is a very little literature available on the quality of model produced in a variability modeling techniques, Both in feature modeling and function-means tree nothing is said on how to validate a particular model, it is suggested to apply the general model qualities like usability, formality and complexity but it completely depends on the situation in hand [1]. Configurable component modeling, on the other hand, provide validation square [14] as validity criteria whose purpose is to addresses the issues like verification (accuracy and the predictive power of theories, methods and models) and validation (internal consistency and external relevance) of engineering design research.

Information Detail –

A variability model must hold the information regarding the entities being used in it. Feature model use an attribute list with each of features or entities which contain information about different things as described in section 2.1.4. Function-means approach emphasize on complete information and suggest to maintain and use Olsson tables along with attribute list of each mean object and to link these tables with function-means as shown in case studies. Configurable component modeling approach uses information model which includes all relevant object types and their relations. This has been shown in detail in case of automotive case study (figure 15) where configurable component is linked to design parameters of function means.

Feature models provide middle level information, function-means tree provide well structured approach for keeping information, and configurable components hold the most detailed information as compared to other two techniques.

Integration with External Processes –

Integration of the variability modeling approach with external process can be very helpful to build a variability model of a product, and to get benefits of previously gained knowledge it is very helpful to integrate it with the product development process.

The authors in [1] have given an example of integrating the feature modeling into the SPICE. Function means tree can also be possible to use with some sort of external development process. The author in [14] suggests that, although, many external processes can be used with configurable component, there is not a single specified method. Some of the methods have been shown in section (application of cc).

It is possible for all of the modeling techniques to be used with external processes of development.

Choices –

All the variability modeling techniques, in fact, employ either of the two different ways to model choices. Either structure of the product is in the mind and features or components (variability) are first-class entities, OR main concept of the model in variability and choices are the first-class entities. These two are named as multiplicity in structure and choices model respectively [15]. Most of the techniques use multiplicity in structure to model variability. Main entities in their models are feature or components/objects. Both the feature modeling and function-means tree use multiplicity in structure i.e. the focus of capturing variability is from the structure perspective of the product, while configurable component modeling approach uses multiplicity in choices.

Abstraction –

To manage high number of variants and large product families variability, modeling techniques use the way abstracting things by introducing either hierarchy or layered models or both of these. Feature modeling use the hierarchy in a tree like structure by abstracting the top level features and then sub division starts toward the concrete small parts or features. It also support layered model along with tree hierarchy. Function means also use abstraction by putting the function means in a hierarchy, the layered model can also be used although we have not found any example such like that in literature but nature of the function-means tree suggest that it is possible. Configurable component uses variant bill-of-material approach and abstraction is possible due to hierarchical nature [14]. Variant bill-of-material includes two types of variants; (i) on the top level, variant control structure is present which is used for selection between different available variants (ii) on the bottom level, static structure (tagged with one or more list of feature labels) is present. A static structure is hierarchical combination of different configurable components. For each leaf in the variant control structure, there is a static structure.

Formal Constraints –

During the modeling of variability the restrictions on the decisions for choices are referred to as constraints here in this paper. Restrictions may be for example the compulsion of selection of a product part when another corresponding part is already selected.

The feature model names these as mandatory, optional or alternative features, while function-means define constraints as optional or compulsory through the out coming constraints in the whole of design process. Object constraint languages are declarative languages which can be used to describe rules for models and can be applied to both feature models and function-means tree.

Conditional existence of variables has been checked in the context of generalized constraints networks (uses *if*<*condition*> *else* while selecting parts) in configurable component modeling technique [14].

All the three techniques have set of formal constraints which help to model and decide upon choices.

Quality Attributes –

Product quality attributes have basic role in success of a product family. A little effort has been made so far in most of variability modeling techniques. In feature models quality attributes such as performance, security and maintainability can be found especially for managing variability in software product lines. In function-means tree the required standards or regulations are taken into account during specification modeling but this is not visualized in the model. In configurable component modeling technique completeness and consistency aspects has been taken into consideration as depicted by the author in [14].

Reasoning Style –

Reasoning style used while the process of modeling the variability is important to get the idea of what this techniques is actually based on and how it proceed towards the actual modeling. Feature modeling uses the inductive approach and style of reasoning in real modeling i.e. in inductive it goes from already built product family towards model of that, while function-means use the deductive technique of modeling the variability, means it is tries to model a product from scratch. Configurable component modeling approach uses abductive approach which was originally performed as a form of action research(between researchers and practitioners in the industry). In abductive approach the reasoning is done from particular to particular, that is, a particular case study is taken and a particular result is generated

Tools Support

The usefulness of a variability models in real life depends on its verification and measurement by the accompanying tool-suite. A tool-support deals with the creation and maintenance of variability models as well as the use of the models [15]. Unfortunately most of the tools available in here are those developed by researchers themselves as prototype just to illustrate their findings of research [1]. The need of mature commercial tools is very strong in the fields of variability modeling. In the following paragraphs, a short description of each of tool used for each of three variability modeling technique is given:

Pure::variants is a commercial tool used for the feature modeling developed by the pure-systems.

METIS is a family of client and server products for creating, visualizing, changing, sharing and managing visual enterprise models. In addition to providing general modeling mechanisms and primitives, Metis doesn't restrict the modeling to one particular methodology. It provides the opportunity for developing in several modeling languages. This tool is used by the [13] to illustrate his work in function means tree, called by him computer specification model.

iMAN PDM System is commercial PDM (product data management) system iMAN – implementation of the information manager (now evolved to TeamCenter Engineering) from Unigraphics Solutions was used to implement configurable components. The choice of software was based on company preference rather than one desired, but the software provided very good functionality for managing configurable components [14]. iMAN is an Internet-enabled product data management system that allows companies to improve their product development processes by organizing, managing and communicating information throughout the product life cycle. It provides set of applications including data management, process management and configuration management tools on an advanced Web-centric architecture.

Some characteristics of the tools described above, associated with variability management can be considered as follows, all the assertions presented here regarding tools, are based upon the tools manuals:

Views –

Views support is different in different tools and varies from multiple views to providing no separate views at all. In feature modeling Pure::Variants provides four different views for modeling and configurations, on the other hand although the Metis support multiple views but the fact is that this is not basically variability modeling tool to restrict the model to be visualized on its best description. iMAN also supports the multiple views.

Active Specifiction/Incosistency –

Active specification inconsistency means that reducing the possible actions that can be taken while creating and maintaining a particular model [22]. There are two types of specifications according to [22], Reactive and proactive. Proactive means tool suggest or prevent certain actions, while in reactive tool only informs user of the inconsistencies.

In Pure::Variants, active specification is available but no inconsistency checking is possible yet. On the other hand while Metis is not actually for variability modeling so it does not give any functionality like this.

Inconsistency can be controlled using active specification i.e. reducing number of actions while creating or maintaining a model. Using iMAN most of the actions are performed automatically.

Configuration Guidance –

The display of range of choices provided regarding the product family by the tools is called configuration guidance.

In both Pure::Variants and Metis the engineer s have to develop their own strategy to develop for making decisions in tree shaped variability model, e.g. breadth first or depth first. These tools do not provide any help with respect to which order should be followed.

iMAN provides configuration guidance to some extent in order to deal with increasing product complexity.

Inference –

Inference engine helps to take decision based on constraints and previous decisions taken in the similar situation. Pure::Variants provides the facility of decision making and consistency regarding model while Metis do not.

iMAN uses inferences to take some decisions e.g. requirement modeling from word document.

Effectuation –

Effectuation is mapping of decisions into actual product family artifacts. It is very important for tools to be useful to provide this type of functionality. Pure::Variants provides the facility of creating the actual product by using XML transformation engine.

Metis on the other hand provide the facilities which are available in a general modeling tool, but not specific to variability modeling.

iMAN provides the effectuation as well.

5.1.1 Comparison Summary

Table 1 summarizes the above mentioned comparison of characteristics in the light of theory and case studies:

Table 1: Comparison of variability modeling techniques

	Characteristics	Modeling Techniques		
		Feature Modeling	Function-means Tree	Configurable Components
Domain	Primary application domain.	Already build systems with high number of variants. Mass products Large bash products.	Appeals more towards product innovation, basically for software product families. Used in so called computer specification model to capture requirements precisely.	Also used for systems with high number of variants especially Platform-based product development in automotive industry - car
	Flexibility	Is flexible to be applied to any product family, but suggested not to use in continuous process.	Can be applied in any product family and situation with some of the customization in abstractions and process.	Highly flexible
Modeling	General development process	Literature review and the identification, classification, modeling and definition of features.	Gathering of top level functional requirements, relevant constraints and design solutions, again sub functional requirements and so on	Inclusion of product variants in the earlier phase into the development process

	Basic Constructs	Features with hierarchical nature and multiplicity notations	Function means in hierarchical tree like nature with attribute list and Olsson table	Configurable components: function-means tree (optional), configuration rules, parametric network, part definition, parameter interface, composition set
	Validity criteria	No formal defined criteria, but suggestion can be found in literature. Major quality attributes are usability, formality & complexity	No formal defined criteria, suggestion are available in literature to follow the traditional quality of models procedure to validate	Validation square
	Information detail	Middle level of information and external references	Good level of information detail and well documented linked knowledge.	Extended information detail using information models
	Integration with external process	Can easily be integrated. Examples are available like SPICE.	Can be used in conjunction with product development.	Can be used with external processes
	Choices	Multiplicity in structure, (feature tree and options)	Multiplicity in structure(functions tree) and function mean objects	Multiplicity of choices
	Abstraction	Hierarchy and multiple layers	Hierarchy and multiple layers	Uses variant-bill-of-materials approach
	Formal constraints	Include/Exclude	Include/Exclude	Generalized constraints network

	Quality attributes	Not modeled	Not modeled, but entries are put into Olsson table of standards, restrictions	Mainly deals with completeness and consistency
	Reasoning style	Inductive	Deductive	Abductive
Tool Support	**Major Tools**	Pure::Variants	METIS	iMAN PDM System
	Views	Pure::Variants 4(modeling and configuration)	1(Two dimensional relations can be shown)	Multiple views
	Active specification inconsistency	Active and no consistency checking is possible	Active	Consistency checking is possible
	Configuration guidance	None	None	Possible
	Inference	Consistency, Decision making	Nothing	Inference facility is used
	Effectuation	File-based operations.	Just as other models can be saved	Effectuation is possible

5.2 Comparison Conclusion

Table 1, summarizes the comparison among three variability modeling techniques on the basis of defined characteristics and case studies in the light of theories and examples. It is clear that these techniques have commonalities as well as differences. The major differences are in the area of primary application domain, basic constructs, and development process. There are commonalities as well, like in choices, abstraction, formal constraints and information detail of models. Each technique covers a particular area of information capturing and variability modeling in best and these has pros and cons.

A careful analysis has leads us to the following conclusions:

- Feature models and function-means trees can be applied to every case study, but they capture different information detail from a little different perspective. For example while implementation of case studies it was obvious that feature models capture product structure, feature models capture functional specifications and configurable components capture from all of these perspectives.

- Feature models seem to be best suitable for capturing product structure along with variability.

- Function-means tree capture requirement specification along with product structure and variability.

- Function-means tree are more appealing to product innovation or redesign rather that product variability management, as the most emphasis is on capturing the functional specifications with the realization of specifications into design. This was verified by the implementation of case studies

- Configurable components have similarity with feature models in many aspects like both are used to manage variability, abstraction, formal constraints etc., but configurable component covers some limitations of feature models as well, like information detail, validity criteria, negative impacts of hierarchical nature, and attribute list. Hence, we conclude that configurable components modeling approach is usage of feature models in another form based on established and well-known design theories and methodologies with some additions.

- Configurable components are relatively new and require lot of work to make it more generalized approach in other fields as well along with the automotive industry.

- Configurable components have already integrated function-means tree as an optional element of its internal structure where function-means tree are used for effectuation of parts.

- There is no mature tool available for either of techniques hence whenever changes in architecture give rise to inconsistency problems because variability modeling techniques with tools support cannot adopt them so quickly.

- None of the techniques has been tested on large scale with full potential which it promises.

- The modeling techniques presented have certain level of abstraction and there is lack of traceability between internal (related to customers) and external variability (related to experts) except configurable components but even it has not been fully implemented and tested.

On the basis of the comparison and conclusions above, it can be said that there is a need of integration among these techniques to get maximum benefit in all respects. Section 5.3 suggests some of these integration possibilities.

5.3 Integration Possibilities

Despite the fact, that each technique has some pros and cons, there exist some commonalities as well. Very little effort or attempt has been made so far to integrate or unify these techniques or at least common aspect of these techniques.

From the analysis of literature review, case studies and comparison discussion, we found out that there is lack of integrated approach that work well with largely heterogeneous artifacts in different domains and product lines. Moreover, most of variability modeling approaches lacks flexibility and extensibility.

From the differences and commonalities identified in previous section, we can conclude that there are certainly some kinds of possibilities for integration among these techniques.

Among the possibilities, there can be options to use feature modeling notations in function-means or function-means notations in feature modeling but the configurable component is a big framework that includes function-means tree itself as a component of it. More over configurable components are themselves very similar in nature to feature tree if multiplicity notations of feature models are used.

5.3.1 Possibility No. 1

The first possibility for integration of two of the modeling techniques has already been met [14], that is, an integrated product model of configurable components and function-means tree. This integrated model addresses two issues (1) the full range of products based on a common platform development; and (2) support for analysis iteration in early phases without the overhead workload and costs associated with management of parts.

This integrated model supports product and manufacturing system development especially in platform-based product development. Feature modeling is, in fact, being already used in an advanced shape in the form of configurable components. Although, it lacks the standard notations of feature modeling, it overcomes with its own notations of inclusion and exclusion.

Feature modeling multiplicity notations has been justified in configurable components as follows: abstract and concrete features in feature model (also see section 2.1.2) are like nodes in control structure in variant-bill-of-material approach. The difference here is that for each node in control structure there exists one static structure. The selection of static structure against control structure is done through matching as shown in figure 22.

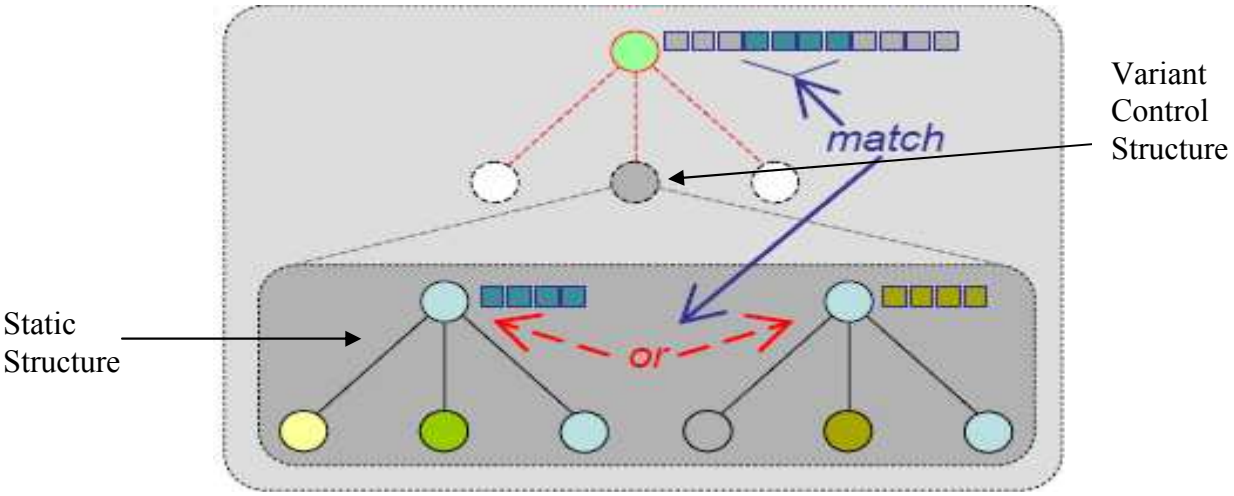

Figure 22: Variant-bill-of-material [14]

Alternative features in feature models are shown in figure 3. A composition element in configurable components is used to choose alternative feature with the help of mechanism to choose alternative. The simplest approach to have default choice and if it is required to change the value, then composition element can request for that component.

Optional features are shown with the help of arcs in feature model while in configurable components optional features can be requested by composition element using configuration request.

The recommendation here is to use feature model notations in static structure as shown in figure 23. Each circle in variant control structure represents a node as in variant-bill-of-material approach. Alternative configurable components are shown with multiplicity 1 or 1..*.

The advantage of using these notations is that in spite of maintaining many static structures, a structure (may be called parametric control structure) can be evolved using feature model notations in the static structure. A critical consideration here is that the feature of evolved structure variant parametric structure must match the attributes with node in control structure. The attributes of variant parametric structure which are used for matching with a node in control structure are to be selected after the evolution of parametric control structure as shown in figure 23.

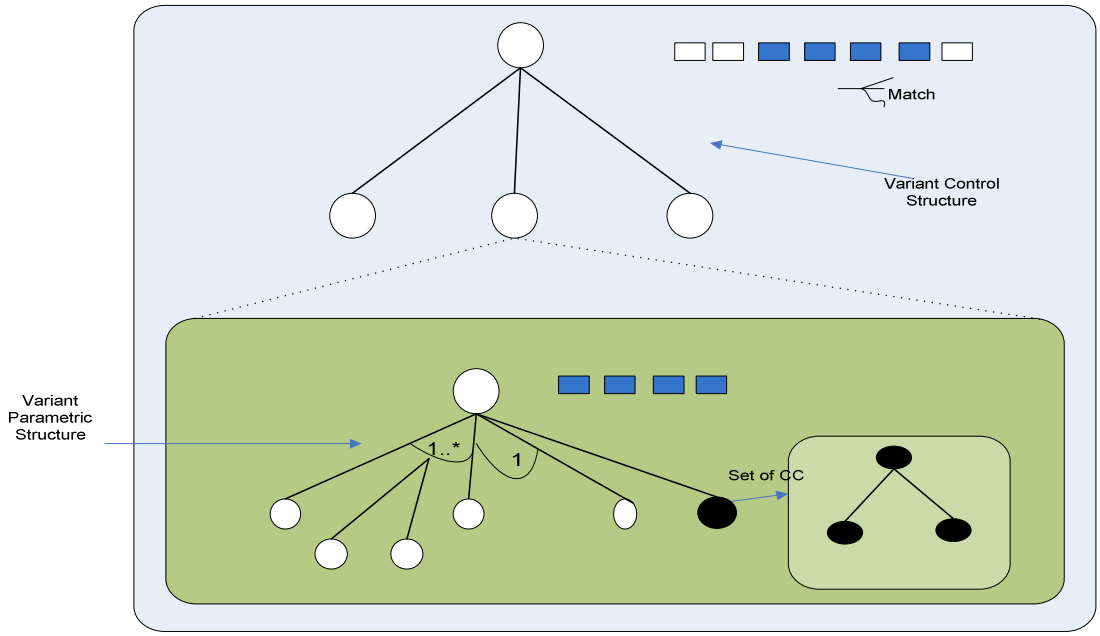

Figure 23: Feature modeling notations in static structure

Compulsory features can be collected to make one configurable components (a configurable component can be set of many configurable components).

Static structures can be formed from variant parametric structure using feature model notations in static structure (now variant parametric structure). These static structures then can be matched with nodes in variant control structure.

5.3.2 Possibility No. 2

Another possible integration is to design all three models in parallel for the same product family and then use cross referencing among them in a way so that each model can access/request every other whenever required. The most promising way is to cross reference those parts which are on the same level of abstraction either from structure or from the functionality point of view.

Example integration is shown in the figure 24. All the three models for each of the techniques are built in parallel and put together in the figure to have a better view. These models were built during the implementation of case study of car door. It is clear from the picture that cars have different systems like engine, door system and electric system and these are shown in hierarchy on the top of figure with rectangles. These systems then have subsystems or modules. One of the subsystems or modules is car door. In case study of car door we have observed that a car door have different structural and functional features or specifications. These were modelled using all three variability modeling techniques and separate models were built for either of the technique. Same models are used here and cross referenced at the top level. It can also cross referenced on the next level of the hierarchy and then at level next to that, but it will make it messy and less understandable. It makes sense to cross reference at subsystem or module level like it has been done here in the figure 24. How to decide on the subsystems or modules is totally dependent on the scenario or situation at hand.

Although all of the potential benefits can be obtained by the development of three models in parallel and then cross referencing them all, but also it will certainly come with redundancy and inconsistency issues. Each of the models will capture common information to some extent, and it will create confusion while maintaining the knowledge base. There may also be a side effect that cross referencing will result into affecting the interdependency within the model itself and thus by disturbing the stability of the model.

All this, may lead this way of integration to wastage of resources and time when it compared to benefits as a careful study is needed in this regard.

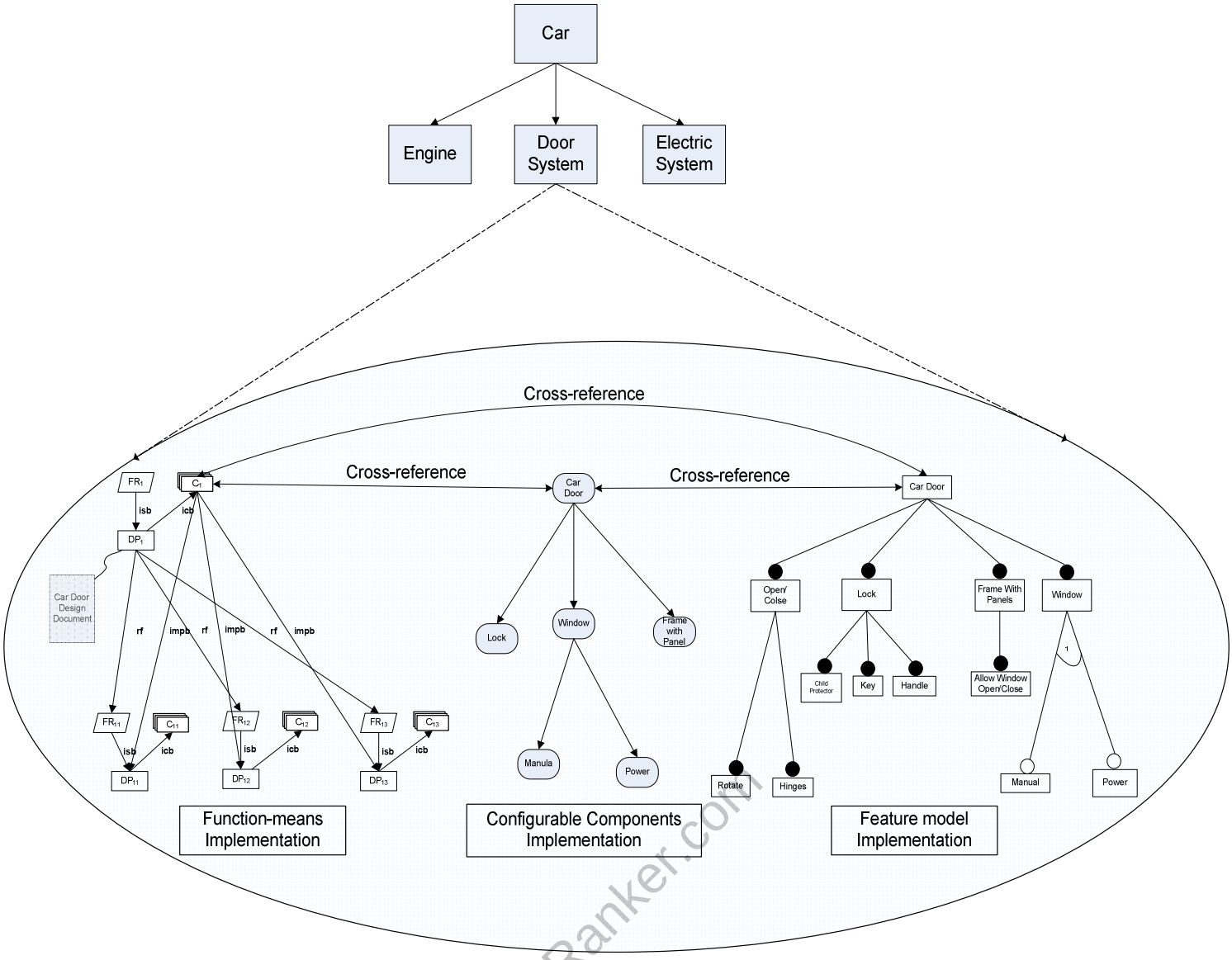

Figure 24: Cross referencing among three vriability modeling techniques

6 Conclusion and Future Work

6.1 Conclusion

The purposed work to compare selected variability modeling techniques has been presented in section 5 along with the integration possibilities. From the results it can be concluded that function-means approach is best suitable for product innovation while feature modeling and configurable component approaches are more suitable for managing variability along with capturing the functional and structural aspects of product families.

The comparison of above mentioned variability modeling techniques showed that although all three techniques have different domain, but if they are modelled for the same reason e.g. managing variability (feature models and configurable components), there are many similarities in their underlying concept for modeling but with different notations. Also these show similarities with the hierarchy and layered abstraction.

It is possible to integrate these techniques into a single technique for capturing the variability, although it has some problems which should be addressed,

An integrated model in section 5.3.1 includes the benefits from all three possibilities at discrete level while possibility in section 5.3.2 includes the benefits from all three possibilities at abstract level.

6.2 Future Work

There are many things which require future efforts such as:

The integration possibilities described in previous section are just based upon the case studies, literature review and educated guess. Although, these integration suggestions have been rooted in literature and careful analysis, these needs to be further explored and implemented in the form of some kind of framework. More over there is strong need for a comprehensive tool to support for these techniques. Further there should be some attempt to implement these techniques on large scale or there should be some research on how much of these techniques are actually being used in industry.

7 References

[1] Christer Thörn, Kurt Sandkuhl, Feature Modeling: Managing Variability in Complex Systems, Complex Systems in Knowledge-based Environments: Theory, Models and Applications 2009: 129-162

[2] Neighbors, J.: Software Construction Using Components. PhD thesis, University of California (1980)

[3] Kang, K., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, S.A.: Feature-Oriented Domain Analysis (FODA) - Feasibility Study. Technical Report CMU/SEI-90-TR-21, Carnegie-Mellon University (1990)

[4] Simos, M., Creps, D., Klingler, C., Levine, L., Allemang, D.: Software Technology for Adaptable Reliable Systems (STARS) Organization Domain Modeling (ODM) Guidebook Version 2.0. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems (1996)

[5] Creps, R.E., Simos, M.A., Prieto-Dıaz, R.: The STARS Conceptual Framework for Reuse Processes. Technical report (1992)

[6] Technical Report STARS-AC-04110/001/00, Paramax Systems Corporation (1992).

[7] Clements, P., Northrop, L.: Software Product Lines: Practices and patterns.Addison-Wesley, Reading (2002)

[8] Czarnecki, K.: Domain Engineering. Technical Report DOI: 10.1002/0471028959.sof095, Encyclopedia of Software Engineering (2002).

[9] Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering 5, 143–168 (1998)

[10] Czarnecki, K., Eisenecker, U.: Generative Programming. Addison-Wesley, Reading (2000)

[11] Riebisch, M., Bollert, K., Streitferdt, D., Philippow, I.: Extending Feature Diagramswith UML Multiplicities. In: Proceedings of 6th Conference on IntegratedDesign & Process Technology (2002)

[12] Riebisch, M.: Towards a More Precise Definition of Feature Models. In: Riebisch,M., Coplien, J.O., Streitferdt, D. (eds.) Modeling Variability for Object-Oriented Product Lines, Norderstedt (2003)

[13] PETER SCHACHINGER and HANS L. JOHANNESSON Computer modeling of design specifications

[14] Anders Claesson,: A configurable component framework supporting platform-based product development. Göteborg : Chlamers tekniska högskola, 2006, 148 s. : ill, 91-7291-791-1

[15] M. Sinnema, S.Deelsstra, Classifying variability modeling techniques, Elsevier Journal on Information and Software Technology, Vol. 49, pp. 717-739, July 2007

[16] Gunter Reichart: Trends in der Automobilelektronik. Presentation at 2. ISST-Forum, Berlin, 28.-29. April 2004; Fraunhofer ISST

[17] Van Deursen, A., De Jonge, M., Kuipers, T.: Feature-Based Product Line Instantiation using Source-Level Packages. In: Proceedings of SPLC2 (2002).

[18] Malmqvist, J. (1997a) Improved Function-Means Trees by Inclusion of Design History Information. Journal of Engineering Design, vol 8(2): 107-118.

[19] Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, H., Pohl, K., 2001. Variability Issues in Software Product Lines, Proceedings of the fourth International Workshop on Product Family Engineering (PFE-4), pp. 11–19.

[20] Zhang, H., Jarzabek, S. (2003). An XVCL Approach to Handling Variants: A KWIC Product Line Example. IEEE Proceedings of the 10th Asia-Pacific Software Engineering Conference (APSEC'03). [18].

[21] Sommerville, I. (2001). Software Engineering. 6th ed., Addison-Wesley

[22] Medvidovic, N., Taylor, R. N., 2000. A Classification and Comparison Framework for Software Architecture Description Languages, IEEE Transactions on Software Engineering, vol. 26, no. 1, pages 70-93.

[23] http://en.wikipedia.org/wiki/Function_Means_Tree (June 01, 2009)

[24] http://www.plm.automation.siemens.com/en_us/products/teamcenter/solutions_by_product/bill_of_material.shtml (May 25, 2009)