NON-FUNCTIONAL VARIABILITY MANAGEMENT BY COMPLEMENTARY QUALITY MODELING IN A SOFTWARE PRODUCT LINE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜR GÜRSES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

NON-FUNCTIONAL VARIABILITY MANAGEMENT BY COMPLEMENTARY QUALITY MODELING IN A SOFTWARE PRODUCT LINE

submitted by **ÖZGÜR GÜRSES** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen Dean, Graduate School of **Natural and Applied Sciences**	____________
Prof. Dr. İsmet Erkmen Head of Department, **Electrical and Electronics Engineering**	____________
Prof. Dr. Semih Bilgen Supervisor, **Electrical and Electronics Engineering Dept., METU**	____________

Examining Committee Members

Prof. Dr. Hasan Cengiz Güran Electrical and Electronics Engineering Dept., METU	________________
Prof. Dr. Semih Bilgen Electrical and Electronics Engineering Dept., METU	________________
Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı Electrical and Electronics Engineering Dept., METU	________________
Assoc. Prof. Dr. Onur Demirörs Informatics Institute, Information Systems Dept., METU	________________
Müge Karaman Çolakoğlu, M.Sc. ASELSAN A.Ş.	________________
Date:	**14.09.2010**

ABSTRACT

NON-FUNCTIONAL VARIABILITY MANAGEMENT
BY COMPLEMENTARY QUALITY MODELING
IN A SOFTWARE PRODUCT LINE

Gürses, Özgür

M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Semih Bilgen

September 2010, 116 pages

Software product lines provide the opportunity to improve productivity, quality and time-to-market of software-based systems by means of systematic reuse. So as to accomplish systematic software reuse, elicitation of commonality knowledge is to be upheld by the analysis and management of variability knowledge inherent in domain requirements. Considerable effort is devoted to the management of functional variability, often neglecting the impact of quality concerns originating from non-functional requirements. In this thesis, a hybrid approach concentrating on the modeling of quantitative as well as qualitative concerns on quality has been proposed. This approach basically aims to support the domain design process by modeling non-functional variability. It further aims to support application design process by providing trade-off selection ability among quality concerns to control functional features that belong to the same domain. This approach is implemented and evaluated on an example domain to reveal its benefits on non-functional variability.

Keywords: Software Product Lines, Variability Modeling, Non-functional Requirements, Software Quality Attributes

ÖZ

YAZILIM ÜRÜN HATTINDA İŞLEVSEL OLMAYAN DEĞİŞKENLİĞİN BÜTÜNLEYİCİ KALİTE MODELLEME İLE YÖNETİMİ

Gürses, Özgür

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Semih Bilgen

Eylül 2010, 116 Sayfa

Yazılım ürün hatları, dizgeli yeniden kullanım aracılığıyla, yazılım tabanlı sistemlerin verimini, niteliğini ve pazara sürüm süresini iyileştirmektedir. Yazılımların sistematik biçimde yeniden kullanımını başarmak için ortaklık bilgisinin belirlenmesi sürecinin, alan gereksinimlerinin doğasında yer alan değişkenlik bilgisinin analiz ve yönetim faaliyetleriyle desteklenmesi gerekmektedir. İşlevsel değişkenlik yönetimi için önemli ölçülerde çaba harcanmakta olup işlevsel olmayan gereksinimlere dayalı nitelik kaygılarının etkisi çoğu kez yadsınmaktadır. Bu tez çalışmasında, hem nicel hem de nitel kalite kaygılarının modellenmesi üzerine yoğunlaşan karma bir yaklaşım önerilmiştir. Bu yaklaşım, öncelikle işlevsel olmayan değişkenliği modelleyerek, alan tasarım sürecini desteklemeyi amaçlamaktadır. Ayrıca, aynı alana özgü işlevsel yetenekler üzerinde denetim sağlamak için, kalite kaygıları arasında ödünleşime dayalı seçim kabiliyeti sağlayarak, uygulama tasarım sürecini desteklemeyi amaçlamaktadır. Bu yaklaşım, işlevsel olmayan değişenlik üzerindeki katkısını ortaya çıkarmak amacıyla örnek bir alan üzerinde uygulanmış ve değerlendirilmiştir.

Anahtar Kelimeler: Yazılım Ürün Hatları, Değişkenlik Modelleme, İşlevsel Olmayan Gereksinimler, Yazılım Kalite Özellikleri

In memory of

Barış Gürses

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Prof. Dr. Semih Bilgen for his guidance, supervision and understanding as the study is evolved from an idea to an implementation. Without his encouragement, this study would be harder to accomplish.

I would like to thank committee members for their valuable contributions on the thesis by means of their comments and discussions.

I owe my deepest appreciation to my mother Fatma Gürses, my father Salih Gürses and my beloved sister Gözde Gürses for their perpetually unconditional love and support.

I would like to thank my colleagues in ASELSAN Inc. for their support and guidance throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AE	Application Engineering
CASE	Computer Aided Software Engineering
COTS	Commercial-off-the Shelf
COVAMOF	ConIPF Variability Modeling Framework
DE	Domain Engineering
DoO	Degree of Orthogonality
DRT	Data Refreshment Time
ECR	Extra Constraint Representativeness
EFM	Extended Feature Model
FIPAP	Feature Interaction Problem Avoidance Percentage
FODA	Feature Oriented Domain Analysis
FORM	Feature Oriented Reuse Method
FR	Functional Requirement
F-SIG	Feature Softgoal Interdependency Graph
HW	Hardware
MMLS	Meteorological Measurement and Logging System
MTBF	Mean Time Between Failures
NFR	Non-functional Requirement
PER	Permissibility Ratio
PL	Product Line
PW	Password
RPER	Permissibility Ratio
RE	Requirements Engineering
RSEB	Reuse-Driven Software Engineering Business
SIG	Softgoal Interdependency Graph
SIMPLE	Structured Intuitive Model of Product Line Economics
SPL	Software Product Lines
SPLE	Software Product Line Engineering
SW	Software
TCO	Total Cost of Ownership
QA	Quality Attribute
UML	Unified Modeling Language
VM	Variability Management

CHAPTER 1

INTRODUCTION

Development of large scale software systems has been a great challenge compelling software developers and software researchers for years. As the software systems demanded become large in scale, it becomes more and more inefficient to develop them from scratch or as a single component. The Software Product Line (SPL) concept has been proposed to build these software systems from several components that can be reused for the development of different software systems with different requirements providing the developers the chance to manage the configuration, in other words the commonality and variability properties of each product developed different from others in the end.

There have been many studies based on the conceptual scope, phases, development and applicability of SPL during the last two decades. As the basic requisite of conceptual development of SPL, domain analysis poses its importance since the construction of variable applications can be realized on a well-defined, mature domain. This reveals the vitality of commonality and variability analysis of the domain being developed as well. Most of the recent studies rely on the expressive and comprehensibility power of features which is a kind of a common language with the ease to understand and agree on by all stakeholders. This is the main reason where the paradigm of Feature Modeling in SPL stems from. In recent studies, the expressive power of the Feature Modeling paradigm is particularly used for modeling variability in terms of characteristics of possible variant products.

Since main considerations for the features of a domain simply originate from the requirements of that domain, there is a need for a clear understanding and explicit elicitation of the requirements involved. There is a strong connection between the requirements and features of a domain. This brings the need for a competent analysis of requirements of a domain if the key to realize this task is feature modeling.

As for analysis of requirements for the sake of explicit description and elicitation of requirements, classification of requirements as Functional and Non-functional helps for further identification. As a basic definition, a functional requirement (FR) is the one that would allow the user or customer to perform some kind of function on the product. A non-functional requirement (NFR) is some kind of a constraint or a restriction on the product that must be taken into account during the design of the solution. These constraints and restrictions may somehow limit users as a result of the interaction involved.

In spite of the complex and vague nature of NFRs, subsequent to their elicitation, somehow these requirements have to be projected on the problem domain. Furthermore as a result of this projection, they have to be incorporated in design abstractions in the form of quality attributes (non-functional features) with suitable approaches for the sake of taking them in account during domain analysis phase. Similar to the experiences with approaches for functional features, quality attributes (QAs) – derived from NFRs - inspire the need of variability management due to the probable need for different levels of their involvement in design decisions. Furthermore, there are complex relationships among different types and levels of QAs in addition to the relationships between functional features and QAs. In order to resolve this variability originating from the hard-to-quantify nature of QAs, trade-offs among each of them has to be managed in a systematic way.

In order to achieve a satisfactory analysis of a domain, the effect of QAs should never be disregarded. On the contrary, their potential for contribution to variability has to be incorporated to modeling of features in a controlled and systematic fashion.

1.1. Purpose of the Study

In this thesis study, a comprehensive survey of literature is realized regarding two main issues; namely the variability concept and modeling of QAs. Furthermore modeling of QA issue is broadened with the answers on how QAs are elicited from respective requirements (technically named as NFRs), and how these elicited QAs are represented in proper models.

Inspiring from two approaches for modeling the QAs in the scope of SPL, namely Extended Feature Modeling (EFM) and Feature-Oriented NFR Analysis; Complementary Quality Modeling Approach is proposed and implemented in conjunction with functional feature model which are constructed on the same illustration example developed for this study.

For a competent and concrete evaluation of the results of the implementation, some metrics are gathered from the literature which try to answer the questions regarding the contribution of QA modeling to the variability concern of domains to be analyzed.

Specifically, Relative Permissibility Ratio (RPER), Variability Factor (VF), Commonality, Degree of Orthogonality (DoO), Extra Constraint Representativeness (ECR) and Feature Interaction Problem Avoidance Percentage (FIPAP) metrics are considered for evaluating the proposed techniques for QA modeling.

1.2. Outline

This thesis document includes six chapters. Chapter-2 defines the basic concepts and sub-processes of SPL dealing with details of the variability and its management in the concept of SPL in addition to some evaluation criteria for the variability assessment of feature models.

Chapter-3 introduces the NFR management concept emphasizing on QA variability management with details and discussions on state-of-art modeling approaches proposed in the literature.

Based on comparisons and assessments realized between these approaches, an approach inspired from a synthesis of EFM and Feature-Oriented NFR Analysis approaches is implemented. Comprehensive discussion on the development process and assessment of these approaches on the selected illustration example is presented in Chapter-4.

In Chapter-5, first, evaluations based on the criteria introduced in Chapter-2 are realized solely on functional feature model developed for this study; afterwards similar evaluations are realized for not only the functional feature model extended with qualitative QAs but also the functional feature model extended with Complementary Quality Modeling approach as the proof of concept. Towards the end of Chapter-5, a detailed comparative discussion based on the evaluations for each model is provided.

As the conclusion of the work, Chapter-6 assesses the whole material in terms of its benefits, drawbacks and contributions to the literature. Besides, possible future work is suggested within the context of this chapter.

CHAPTER 2

SOFTWARE PRODUCT LINES AND VARIABILITY MANAGEMENT

In this chapter, first, an overview of the literature on SPL development process is given. The scope, capabilities, and applications of the process with its sub-phases are summarized. Then one of the fundamental principles of SPL concept, variability management, is introduced with an overview of Feature Modeling approach and the state of art Computer Aided Software Engineering (CASE) tools serving for systematic construction of variability models.

2.1. Overview

SPL is defined as a collection of software-based systems sharing common set of features which are supposed to be managed efficiently in a well defined domain. Various and specific needs of any particular market segment or mission are aimed to be satisfied. The developed systems are aimed to be originated from a common set of core assets in a prescribed way. [1]

The key issue in SPL is software reuse. Similar to mathematicians using the same formulas to solve different problems or physicists using the same laws to explain different phenomena, software system designers are to use these same software-based systems or modules sharing common set of features for the development of various software-based systems feeding the needs of any particular market. [3]

Software reuse serves the potential to increase productivity, improve quality, and reduce risk during the design and development of the software-based systems. But the reuse of software sub-systems or modules is not enough, since the concept of how software is developed based on reuse is usually ignored. This limits the success that software reuse can meet.

In order to get rid of these limits, reuse should be supported with requirements engineering in which the application is to be analyzed for reuse before writing its specifications for reuse. The process is also to be supported with how reusable specifications are retrieved and validated. For further support, domain analysis can serve for the identification of user requirements for reuse. During the design and development phase of software assets, the process has to be supported systematically by the practices above.

2.2. SPL Processes

SPL development consists of two processes that run in parallel, namely, product line engineering based on core asset development (which can also be included in the context of domain engineering (DE)) and application engineering (AE) the aim of which is individual product development. [1]

The DE process includes activities for analyzing systems in a domain and helps in the creation of reference architectures and reusable components. The AE process includes the activities for developing applications using the artifacts (i.e. domain model, domain architecture) created by DE. [2]

Figure-1 below illustrates how DE and AE work together in parallel.

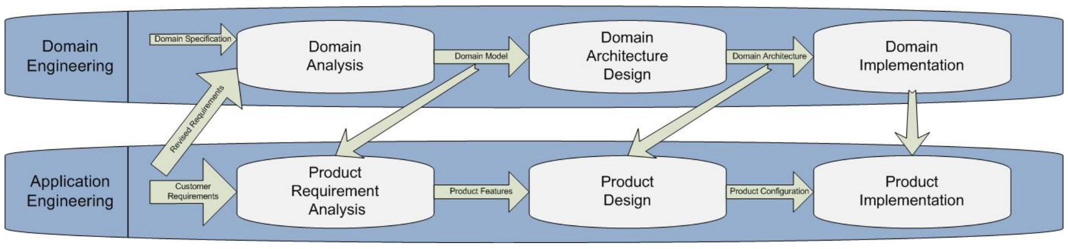


Figure 2.1: SPL Processes (modified from [9])

2.2.1 DE Process

The process in which the development of the product line (PL) architecture is realized can be regarded as DE phase. DE aims to collect, organize and store recent information and experience acquired during building systems or parts of systems in a specific domain in the form of reusable assets. Furthermore, DE aims to provide an appropriate way to reuse the assets while building new systems. [2]

DE consists of three main steps namely as domain analysis, (domain) architectural design and domain implementation. During domain analysis sub-process, the application scope of the whole PL is analyzed with the contribution of requirement analysis for the PL. [9]

For the development of core assets in a SPL, it requires basically the domain analysis which identifies commonality and manages variability within. In the study of domain analysis paradigm, several modeling approaches and analysis techniques are proposed.

Feature oriented domain analysis (FODA) [5] has been established to identify commonalities and variabilities in a domain in terms of product features where the feature can be defined as an abstract communication medium between the customer and the developer having a common meaning for both parties which defines product characteristics. FODA uses this medium in order to identify commonality and variability effectively among different products in a domain. Furthermore it provides a basis for developing, parameterizing and configuring various reusable assets. [1]

FODA is extended into Feature Oriented Reuse Method (FORM) to support architectural design and object-oriented component development for the incorporation into marketing perspective and exploration of analysis and design issues using this perspective.
Several other attempts have also been made for the extension of FODA. Reuse-Driven Software Engineering Business (RSEB) [6], a method based on Unified Modeling Language (UML) notations with the feature model of FODA, was proposed to be used for reuse in object-oriented software engineering. [1] UML [7] has many advantages in providing greater insights into understanding and managing commonality and variability. Using UML notation, the functional requirements view is represented through a use case model, the static model view through a class model, and the dynamic model view through a collaboration model and a state chart view.

In following the domain analysis, SPL architecture is designed which provides the framework for reusable components. Reusable components are designed in the last step of DE, namely during domain implementation. [9]

2.2.2 AE Process

The process in which the development of individual projects over the PL is realized can be regarded as the AE phase. AE aims to develop software products using DE artifacts which guide developers throughout the selection of proper architecture model and existing components. [2]

AE consists of three main phases, namely, product requirement analysis, product design and product implementation during which component integration is realized.

During product requirement analysis, the requirements on each respective product are specified individually in relation with the domain requirement analysis performed in DE. Feature analysis for each respective product is realized. As the feature analysis is performed, product features are selected with the help of customer requirements and domain model artifacts. These features are used for the definition of product configuration that is composed of software components. The architecture of each respective product is derived with the help of SPL architecture. Finally in product implementation phase, product-specific components are implemented. These components are tailored to constitute a software product on the selected domain architecture model. Components are tailored especially for the enhancement of their adaptability through the interfaces in order to realize their integration with others to form the desired final software product.

During definition of partial involvement of any component in overall functionality of the system, there exists a trade-off between the size and functionality of it. Granularity of the components poses its importance in the sense of flexibility and maintainability such that large components reuse more software but are harder to compose and maintain whereas small components might embed too little functionality. [4] According to Chung et al. [8], as software components become larger, their reuse value greatly increases whereas the ease of adapting and integrating them decreases. Therefore a comprehensive effort during the design of the components is needed for the sake of fine-tuning the level of this granularity. Experience gained during the application phase and FRs engineered during domain engineering phase are the main inputs for defining this measure.

Components need to be designed in a substitutable way such that a component could be replaced by another regardless of either at design time or run-time. This provides flexibility during the development of the whole system when AE is of concern.

In order to enhance the reusability effectiveness of a component, a significant effort has to be provided for the thorough documentation, testing and verification facilities of each.

2.3.Variability Management and Feature Modeling

2.3.1 Variability Management

Variability Management (VM) is one of the fundamental concepts in SPLE as the main purpose of SPL is to support variants (different choices) of products by not only taking into account the commonalities but also the variabilities extracted from the domain. For the sake of development and production of a wide range of variant systems from a defined domain, during DE phase, variability has to be explicitly elicited from the requirements thereafter by defining, representing, exploiting, implementing, evolving, in other words managing it throughout all sub-phases of SPLE in all sets of software artifacts constituted from requirements such as architectures and components. [25] As of the most critical sub-phases of VM, variability is said to be defined during DE and to be exploited during AE by configuring appropriate variants.[24]

SPLE offers the differentiation flexibility and diversifiability of end products. As the dependencies, restrictions, relations between different variabilities are managed systematically, this ability of flexibility in terms of diversifiability is guaranteed to be enhanced as the dynamics of diversifiability are kept under control. In order to achieve this enhancement, systematic identification and management of variability has to be supported with appropriate approaches, techniques and tools. [26]

Variability subject involved in any domain has to be represented by proper abstractions namely by variation points. These are the points where differences exist in the final systems. They are the source of different feature possibilities namely the variants existent in the domain to be satisfied. [24][25]

In order to represent variability in a domain, modern approaches use features as basic concept for variability representation.

2.3.2 Feature Modeling

Feature Modeling paradigm was first introduced to the literature in the context of the Feature-Oriented Domain Analysis (FODA) Method by Kang et al. [5] in 1990. The paradigm takes advantage of the features as they are externally visible characteristics that can be utilized to differentiate one product from the others. Additionally, Feature Modeling helps in scoping of product-line by selection of features which are desired to be supported by the PL and which are not. [28] In a poorly scoped product-line domain, relevant requirements as derivation points for features may not be implemented or some implemented requirements may never be used, leading to redundant complexity in addition to development and maintenance costs. [32]

Stemming from the commonality and variability insight of PLs, features as the external visible characteristics of products are easier to identify than the conceptual abstractions (i.e. functions, objects, components, aspects etc.) derived from internal viewpoints. This further supports the participation of all stakeholders during not only production but also development of software modules.

Lee et al. [1] define Feature Modeling as "the activity of identifying externally visible characteristics of products in a domain and organizing them into a model called a feature model". During this activity, features are arranged hierarchically, based on relations specified between parent features (variation points as features) and respective child features (variants as sub-features). There exists several notations for the expressions of relations between the features. The notation proposed by Czarnecki et al. [21] seems to be the most comprehensive and the one that is widely used. These parent-to-child relationships can be specified as follows:

- Mandatory – parent feature requires all its mandatory child features
- Optional – parent feature may include any number of optional child features
- Alternative – parent feature requires exactly one feature from a group of alternative child features
- Or – parent feature requires at least one feature from a group of or child features

It should be noted that a child feature can appear in a product if only its parent feature is included.

These relationships are regarded as structural interdependencies by Jarzabek et al. [20] as they are explicitly defined interdependencies. On the contrary, interdependencies such as relationships of requires or excludes and correlations between features are regarded as implicit interdependencies (also regarded as cross-tree constraints [33]) that are implicit modeling abstractions. A selection of a specific feature may require the selection of any other specific variant, similarly, a selection of a specific variant may avoid the selection of any other specific feature as well, so called "Requires Relationship" and "Excludes Relationship" respectively**.** Main difference between structural and implicit interdependencies can be specified as follows; structural interdependencies are explicit modeling abstractions for guidance especially for feature selection (especially during AE) and they somehow may be visible to customers, whereas implicit interdependencies have an obscure nature and usually they are not visible to some stakeholders (i.e. customers). Pohl et al. [24] defines this difference as external variability and internal variability respectively. Some stakeholders (application engineers and especially customers) do not need to take the implicit one into consideration whereas the domain expert designing feature model has to realize these connections between features. In this respect, implicit interdependencies should be utilized in the models in a controlled manner such that augmenting the usage rate of these dependencies leads to visual complexity as the perception of all relationships thoroughly gets harder.

Not only during the construction of domain architecture but also during building the applications, the utilization of feature diagrams is inevitable. Basically feature diagrams are graphical representations of feature models.

In feature diagrams, the relationships between features are structured in a hierarchical tree format in order to form a suitable feature model and facilitate the feature selection process. These trees are constituted of nodes and directed edges in which nodes are mapped to features and directed edges are used to reveal interrelationships between these features.

During the usage of features for the modeling of variability, the perceptual advantage of graphical notation is utilized in most of the feature modeling approaches as well. Beginning with the emerge of feature diagram concept with FODA, a diversity of graphical feature modeling notations are proposed so far, examples of which are compiled with a comprehensive and comparative discussion by Metzger and Heymans. [35] Originating from FODA, commonly and widely used basic notations for Feature Diagrams can be observed in Figure 2.2 below in which structural and implicit relations are defined among features and their respective sub-features.

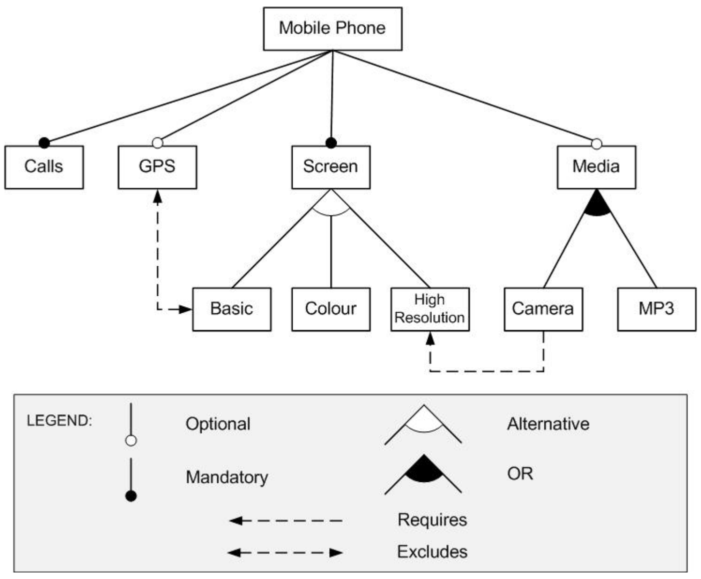


Figure 2.2: An instance of feature model [33]

As an apparent instance for co-utilization of both structural and implicit interdependencies, with respect to the Figure 2.2 above which demonstrates the supported features of a software to be loaded in a mobile phone; all the phones possess support for calls in addition to display support for only one of either basic, colour or high resolution screens. Moreover, the software may optionally possess support for a GPS and one or both of camera and MP3 multimedia facilities. In terms of implicit interdependencies; inclusion of camera support feature in a product automatically implicates high resolution screen support into the product configuration whereas including GPS feature support automatically precludes basic resolution screen support out of the product configuration and vice versa.

Consequently, using this graphical demonstrative power of features, the notion of Feature Diagrams is widely accepted in state of art Feature Modeling approaches.

As clearly specified in [33], there are some issues to be taken into account during the construction of implicit interdependencies (a.k.a. cross-tree constraints).

A feature model is defined to be void if it represents no products. In relation to this specification, main reason that drives a feature model to void is the wrong usage of cross-tree constraints such that a feature model without any cross-tree constraints can never be void.

Another possible outcome of misusage of cross-tree constraints is the dead features as such features cannot be included in any of products due to wrongly defined interrelationships including structural ones. These relationships are needed to be avoided as they give wrong idea regarding the expression of the domain. Examples of such situations are demonstrated in Figure 2.3 below. [33]

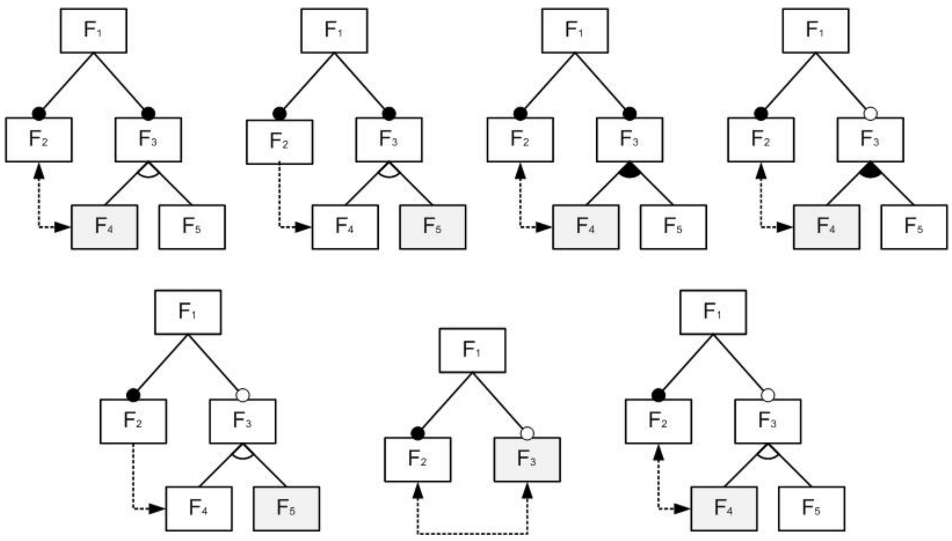


Figure 2.3: Examples of dead features (Shaded features represent dead features) [expanded on [33]]

With relevance to dead features issue, one other outcome of wrong cross-constraint usage is the false optional features. False optional features are the ones that are included in all of the products although they are not modeled as mandatory features in the feature model. [33]

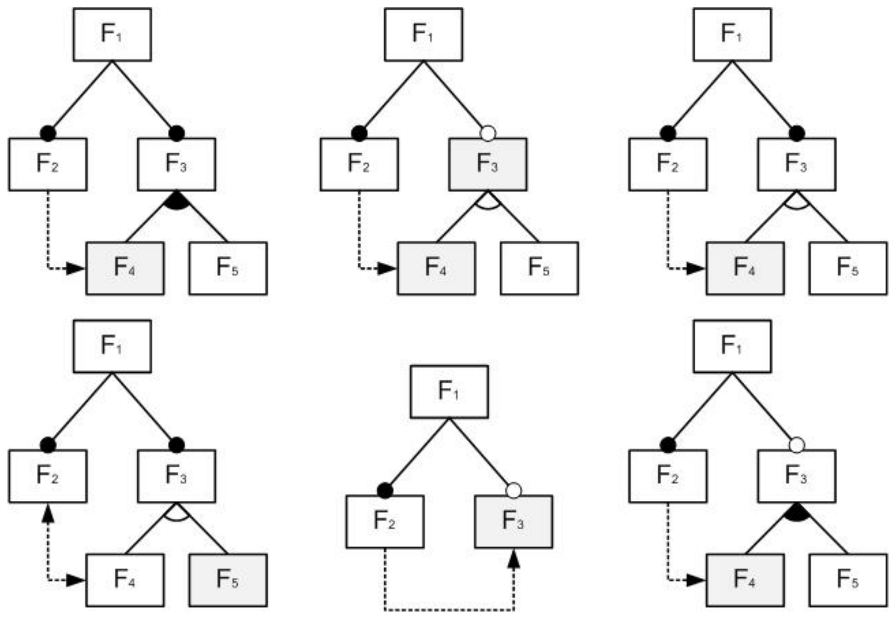


Figure 2.4: Examples of false optional features (Shaded features represent false optional features) [expanded on [33]]

Constraints (namely the implicit interdependencies) between features are essentials for the establishment of a substantial feature model due to the need of following facilitations: the modeler (possibly the domain expert constructing domain architecture) may desire to state that a feature F_1 requires a feature F_2 to perform its functionality properly leading to a situation that if application engineer selects F_1 for an application, F_2 has to be selected as well. In a similar manner, the domain engineer may desire to state that if a feature F_1 can not perform its functionality with the presence of feature F_2 leading to a restriction for the application engineer to select F_2 if feature F_1 is selected previously. In the light of variability modeling, all these possibilities of relationships are classified by Pohl et al. [24] to be among variant-to-variant, variation point-to-variation point and variant-to-variation point as conventions included in the concept of Orthogonal Variability Model where variation points can be regarded as features and variants as sub-features.

As a consequence of conjunction in between these structural and implicit interdependencies, in terms of scalability of the variability knowledge involved in the model, there is an important need of control over trade-off among the structural and implicit interdependencies. This is due to the fact that the structural one is the driving force for variability whereas the implicit one put essential limits around variability as the rules it defines are used to verify consistency and completeness of not only the features but also the entire model.

2.3.3 Tool Support for Management and Analysis of Feature Models

Being the most widely used variability modeling and management mechanisms, the support of CASE tools is required by Feature Modeling approaches for the systematic management of the modeling knowledge. They need the visual expression ability during the development and usage of this knowledge and select them as a variability mechanism in the end.

A number of feature modeling tools have been proposed and reported in the literature: XFeature [27], FeaturePlugin: Feature Modeling Plug-in (also known as fmp) [28] , FeatureIDE [29], Captain Feature [30] and the commercial tool pure::variants from pure-systems GmbH [31].

Different from the feature modeling tools referenced above, a framework named FAMA (Feature Model Analyzer) is proposed [40] in order to realize automated analysis of feature models. The framework is available to be implemented on Eclipse Platform [41]. As long as a feature model expressed in XML is provided to the tool, analysis of the feature model can be performed with the help of the most widely used solvers (i.e. SAT, CSP and BDD) in the literature. These solvers are utilized to analyze feature models which are expressed in the form of either of Boolean Satisfiability Problem (SAT), Constraint Satisfaction Problem (CSP) or Binary Decision Diagram (BDD). The reasoners included in the framework implementation are capable to answer many of questions related with the characteristics inherent in the feature model. Two of the primary facilities upheld by the tool are providing the total number of possible products of a feature model and the total number of products that contain a specific feature.

In the context of the goals and scope of the present study, it can be observed in *Chapter 5* that FAMA is utilized for evaluating the benefits of the proposed QA modeling approach. FAMA provides essential parameters needed by several types of evaluation criteria specific to different models of comparison.

2.3.4 Evaluation Criteria for Feature Models

Various methods and approaches are being implemented on the basis of Feature Modeling fundamentals. In order to make an assessment regarding any approach and its implementation, especially in terms of its variability related contributions, some evaluation criteria have to be defined.

First of all, in order to have a general idea about the complexity and flexibility of a feature model, the number of potential products has to be measured. It is usually accepted that the more the number of potential products derived from a model, the more its complexity and flexibility. A huge number of potential products may constitute a more flexible SPL while leading to more complexity. This measure can be assessed with respect to a Feature Model with or without the relations and dependencies defined between the features. As the relations (i.e. requires or conflicts) are taken into account, the number of potential products is expected to be decreased, which is more realistic in terms of evaluating the complexity and flexibility of the feature model. Below, some specific metrics considered relevant in this context will be reviewed.

2.3.4.1 Relative Permissibility Ratio (RPER)

As an indicator of how well a variability modeling approach fits the need of attaining more permissible systems, Kasikci and Bilgen [38] proposed permissibility ratio (PER). PER is defined, as in expression (1), as the ratio of the number of systems that are acceptable as valid systems by the experts and users of the domain (N_a), to the total number of possible systems that can be generated using the particular modeling approach (N_t).

$$PER = \frac{N_a}{N_t} \qquad (1)$$

Since it requires too much effort to determine the true number of acceptable valid systems by the users and experts (N_a) and due to the difficulty in defining common means of filtering rules to specify N_a, PER is utilized by the criterion of relative PER (RPER). As defined in expression (2) below, RPER is obtained by the ratio of PER of two different variability modeling approaches. RPER is meaningful under the following restrictions:

- the same set of criteria is used by the experts and users for the validation of acceptable systems which leads to the same values of N_a for different variability modeling approaches,
- the models of comparison need to express the variability in terms of the same modeling artifact, namely the features,
- the variability items (i.e. number of features – leaf nodes – in feature model) utilized for the specification of acceptable systems are the same

RPER is obtained as the ratio of the number of possible systems that can be generated using the particular modeling approach where constraints and relationships between the artifacts are taken into account.

$$RPER_{A,B} = \frac{PER_A}{PER_B} = \frac{Nt_B}{Nt_A} = \frac{\text{Number of products belonged to Model B}}{\text{Number of products belonged to Model A}} \qquad (2)$$

As $RPER_{A,B}$ gets closer to zero, the variability modeling capability of model B is said to represent actual realizable systems more accurately.

2.3.4.2 Variability Factor (VF)

In relevance to the discussion above, the variability of a Feature Model is strictly related with the relations and dependencies defined amongst the features in the model. During the description of a Feature Model in terms of relationships between the features, the variability of the model can be said to be described concurrently and indirectly. In other words, the variability depends on relations and its types as these relations restrict the number of potential products. In order to have a measure regarding the variability of a model, as defined by Benavides et al. [13], the Variability Factor (VF) quantifies the ratio of the number of potential products with the feature relations defined to the number of potential products without the feature relations defined as illustrated in (3). This factor has the range of values from 0 to 1 and the more it is close to 1, the more the Feature Model is said to have variability. The value of this factor gives the Feature Model developer the idea regarding the degree of variability that the model possesses. Besides, this factor exhibits the flexibility of the feature model.

Alternatively, for the calculation of VF, the denominator can be assumed to converge to 2^n, where n denotes the number of leaf node features such that the most flexible feature model would be the one that has all its features as optional. [33]

$$VF = \frac{\text{Number of products [with relations defined]}}{\text{Number of products [without relations defined]}} \qquad (3)$$

2.3.4.3 Commonality

It is harder to have an assessment regarding the overall commonality measure of a Feature Model. However, inspiring from the metric utilized by Fernandez-Amoros et al. [32], it is possible to make an assessment regarding the commonality of a feature which can be measured by having the ratio of the number of possible products having a specific feature to the number of the total number of all potential products, wherein the relations between the features are all defined.

Similar to the approach above, Benavides et al. [33] extend the calculation by evaluating with regards to not only a single feature but also a configuration of features (additionally defining the features not to be selected). In reference to (4) below, the evaluation is performed by calculating the number of products that employs a specific feature or a specified feature configuration divided by the number of all potential products derived from the feature model**.**

$$Commonality = \frac{Number\ of\ products[with\ specified\ configuration]}{Number\ of\ all\ products} \qquad (4)$$

The metric spans the values from 0 to 1. The value of this metric can be utilized for the prioritization of the order of the features that are going to be developed since most common features forms the backbone of the referenced PL domain and needs to be developed prior to others. [13]

2.3.4.4 Homogeneity

Conceptually relevant to the commonality measure discussed above, Clements et al. [34] proposed the homogeneity metric implemented on their general-purpose business model called Structured Intuitive Model of Product Line Economics

(SIMPLE). It supports the estimation of the costs and benefits in a PL development organization. The homogeneity metric is proposed to reveal an indication of the degree to which a PL is homogenous (i.e. how similar are the SPL products), considering the fact that not every product exhibits the same commonality.

For instance a more homogenous feature model possesses less unique features in a product whereas a less homogenous one possesses more unique features keeping in mind that unique features can be included in only one product. [33]

The metric has the range of values from 0 to 1, where 0 implies that all the products are unique. As the homogeneity converges to 1, the products are said to be more similar with each other. Inspiring from the equations defined in [33] and [34], homogeneity can be calculated by (5) below where F_U stands for the number of unique features in one product, and NP stands for the total number of different products represented by the feature model:

$$Homogeneity = 1 - \frac{F_U}{NP} \qquad (5)$$

The metric helps for the assessment of PL scoping such that as the evaluation on homogeneity of the products derived from a PL is performed, the degree of reuse among the products in the PL can be derived. This would provide an anticipation and comparison for the levels of component reuse in different PLs.

2.3.4.5 Degree of Orthogonality (DoO)

Czarnecki and Kim [36] introduce the DoO, defined as the ratio between the number of products evaluated by including the effects of all dynamics of the whole feature model (i.e. all the constraints among whole features of the feature model are taken into account) and the number of products in a sub-tree including the effects of local dynamics (i.e. only local constraints in the sub-tree are considered during the calculation of products derived from the sub-tree) as referenced in (6).

$$DoO = \frac{Total\ number\ of\ products\ of\ Feature\ Model}{Number\ of\ products\ of\ Sub-tree\,[with\ local\ constra\mathrm{int}\,s]} \qquad (6)$$

This metric has the range of values from 0 to infinity. A high DoO implies that decisions about feature selections can be realized locally without taking their influence on choices in other parts of the feature hierarchy into account.

2.3.4.6 Extra Constraint Representativeness (ECR)

Mendonca et al. [37] define ECR as the ratio of the number of variables possessing implicit interdependencies (repeated variables counted once) to the total number of variables in the feature tree as referenced in (7) below.

$$ECR = \frac{Number\ of\ features[involved\ in\ cross-tree\ constra\mathrm{int}\,s]}{Total\ number\ of\ features} \qquad (7)$$

The metric encompasses the values between 0 and 1. This measure gives an idea regarding the usage intensity of cross-tree constraints with regards to structural ones. Moreover, the measure can be regarded as the utilization ratio of implicit interdependencies to structural ones. In the light of this idea, as discussed in *Section 2.3.2 Feature Modeling*, while opining on scalability of the variability knowledge involved in the model, this measure would give an idea for the sake of acquiring a comparison and trade-off exercise between these two essentials of variability modeling.

2.3.4.7 Feature Interaction Problem Avoidance Percentage (FIPAP)

In a fashion similar to the evaluation of ECR, Kasikci and Bilgen [38] proposed a metric to measure how better a variability modeling approach avoids problems stemming from feature interactions. As the number of features in a model increase, if the relationships of excludes are not defined sufficiently and appropriately among the features, coexistence of some features may hinder the proper operation of the product. Therefore, it can be deduced that the more exclude relationships exist in a feature model, better the modeling approach is in avoiding Feature Interaction Problems. As defined in (8), relative FIPAP of model A with respect model B is measured based on the exclusion relationship numbers for model A (En_A) and model B (En_B).

$$A_{A,B} = \frac{En_B - En_A}{En_A} x\,100 \qquad (8)$$

CHAPTER 3

NON-FUNCTIONAL VARIABILITY MANAGEMENT AND QUALITY ATTRIBUTE VARIABILITY MODELING

3.1. Non-functional Requirements

The specifications and approaches mentioned so far address formation of components regarding only functional properties or requirements of software systems. As the common drawback of recent approaches, there exist a lack of traceability from NFRs to design and implementation. Not only to ensure confidence in the system but also to achieve successful configuration of components, the effect of non-functional properties - also specified as extra-functional properties or quality attributes in the literature as well - has to be taken into account.

There seems to be no consensus about the nature of NFRs and the way to document them in requirements specifications. Jacobson, Booch and Rumbaugh defined NFR as "*A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement*." [12] Glinz concludes his study regarding the definition of NFR as "*an attribute of or a constraint on a system*." where he defines specific quality requirements being attributes of the system with instances of Reliability, Usability, Security, Availability, Portability, Maintainability etc.[11]. Depending on the context and the application of the system, the set of NFR instances can be extended.

3.2. NFR Analysis

As it is the phase during which requirement analysis for the systems is realized, the domain analysis in SPL development has to consider both FRs and NFRs.

Reasons why NFR analysis during SPL development is necessary can be specified as follows: First of all, different software applications sharing common functional properties may require different levels of NFRs such as reliability, security, privacy, performance, etc. Besides, if NFR analysis is not realized properly, redundant efforts on design and implementation of some NFR aspects may have negative effect on other NFRs of the product such as higher cost of development and maintenance.

It is also more difficult than FRs to analyze and define NFRs due to very vague nature of them together with their variations involved. Besides, most NFRs are closely tied with FRs leading to considerable difficulty to distinguish them from each other. Furthermore, non-functional variations, details of which are to be considered during feature analysis, arise from NFR tradeoffs making them more confused.

3.3. Non-functional Variability Management

Variability is a significant concept in SPLE that has to be managed systematically. There are many different methods and notations focused on functional variability. Despite the complexity of non-functional variability management in SPL due to the ambigious nature of QAs (non-functional features), success in feature-oriented approaches for functional variability modeling has guided recent studies, to be reviewed below, through similar approaches in non-functional variability modeling.

In extension to *Section 2.3.2 Feature Modeling*, during analysis of features as the indispensable phase of DE, the concept of variability modeling needs to be broadened with the QA concerns. As a natural consequence of that, in course of defining relations (i.e. structural and implicit interdependencies) between features in a feature model, non-functional features are to be separated from functional ones, treated in a different way, due to their distinctive nature in terms of their interplay among each other and with functional features. From RE point of view, with expansion on what Jarzabek et al. [20] proposes as classification of interdependencies involved among all functional features and QAs, Correlation Taxonomy of Requirements is illustrated in Figure 3.1 below.

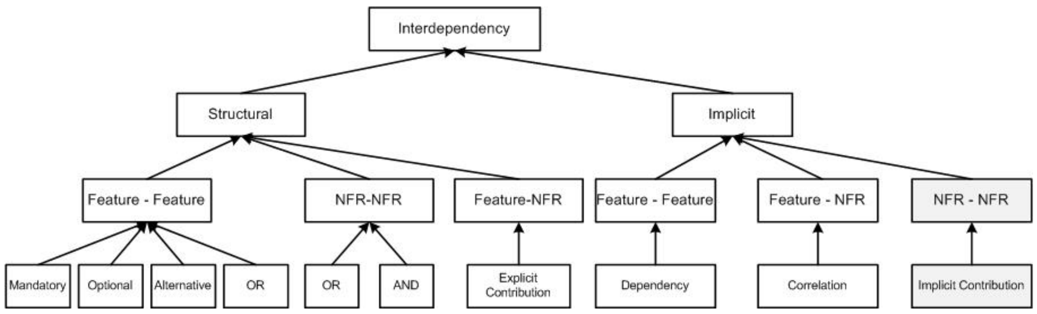


Figure 3.1: Requirements Correlations Taxonomy (expanded on [20])

It should be noted that, during the construction of interrelationships among whole set of features; QAs involved in structural interdependency are proposed to possess OR and AND relations and structural interdependency between a functional and non-functional feature takes explicit contribution form. Besides, implicit interrelation between a functional feature and non-functional feature is to have the correlation relationship which is in the form of either positive or negative (i.e. the choice of a sorting algorithm (functional feature) may influence time performance (QA) in a positive or negative way). As an extension on Jarzabek et al.'s taxonomy, implicit interdependencies among NFRs has to be taken into account during domain analysis since QAs involved in a domain model

encapsulates inexplicit interplays among each other that needs to be observed and controlled. These interplays are possibly in the form of positive and negative impact among each other that we can name as “Implicit Contribution”.
The need for a proper representation technique for the illustration of variability modeling of non-functional features makes the usage of feature model diagrams indispensible in such a domain where it is considerably difficult to capture all the information.

Below an overview of non-functional variability modeling approaches with the particular usage of feature model technique is presented with comparisons.

3.3.1 Extended Feature Model (EFM)

Benavides et al. [13] proposed an extension on classical feature models with extra-functional features and improvement on previously proposed vague notations with the help of allowing relations amongst attributes. Attribute is specified as any characteristic of a feature that can be measured in a defined attribute domain. In relevance to these specifications, extra-functional feature (non-functional feature) is specified as the relation between one or more attributes of a feature. Attributes can be utilized to specify extra-functional information such as cost, speed, RAM memory or development time required to fortify the features as referenced in Figure-3.2 below. These attributes can hold values of a specified range belonged to a discrete or continuous domain (i.e. integer or real number domains). EFMs can also offer complex constraints among attributes and features such as “If attribute A_1 of F_1 is lower than a value X, then feature F_2 can not be part of the product.” [33] which can be modeled in the form of implicit interdependencies amongst features.

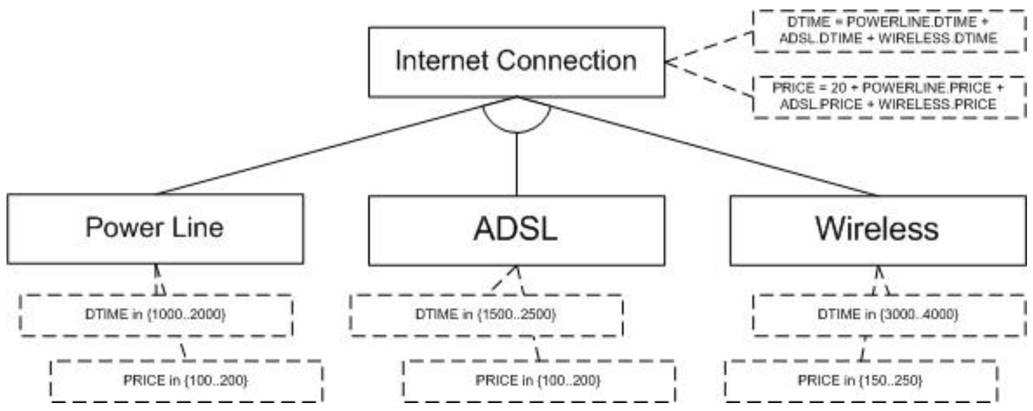


Figure 3.2: Instance of EFM (adapted from [33])

Additionally, a notation inspired from Streitferdt et al. [42] is adopted for the sake of illustrating how attributes decorate several features. It should be noted that the parent features are decorated with functional expressions that are dependent on the attribute values of their child features.

As several attributes are added to features, feature models can be utilized with optimization operations in order to select a set of features that either maximizes or minimizes any value of a given feature attribute. [33]

An algorithm based on Constraint Programming with some definition of rules is proposed in the study which details these relations between the attributes of the classical functional feature model. Furthermore filtering rules are defined for the limitation of potential products of the model having the desired configuration of the user. Besides, validation and optimization rules are defined in the algorithm in order to find out the best products with control over previously specified constraints regarding the derived NFRs. [13]

3.3.2 Definition Hierarchy

Kuusela and Savolainen [14] proposed a definition hierarchy method for the organization of requirements into definition hierarchy where requirements for different products are demonstrated in the same hierarchy. Different from the contemporary design methods, the method analyzes the requirements in two categories as design objectives and design decisions instead of emphasizing only on FRs based on customer's needs.

The hierarchy is structured on a logical AND tree in which the topmost nodes are design objectives (i.e. architectural drivers, other QAs the system has to achieve) with the remaining nodes left as design decisions. Each node in the definition hierarchy has a priority that reflects the importance of it with a clear intention of its parent. The hierarchy is organized in a way that an edge between a design objective and a design decision indicates that the requirement is (partially) satisfied by design decisions.

Definition Hierarchy method aids the designer in management of requirements and their interrelations by resolving the requirement conflicts and inconsistencies with the definition of design decisions and objectives. It helps the designer on the way of finding missing requirements by reverse-tracing the hierarchy tree. Besides if a requirement is considered to be ambigious, it is easier to define it more accurately by adding new nodes of design objectives on the suspected sub-tree.

3.3.3 COVAMOF

Most of the contemporary software variability modeling methods aims to represent variation points as the primary entity with a clear hierarchical organization. The framework for modeling variability in software product families named as COVAMOF (ConIPF Variability Modeling Framework) [15] aims to represent dependencies by modeling their relations as well. The framework uses the CVV (COVAMOF Variability View) for modeling of the artifacts on all abstraction layers of the product family with Variation Point View and Dependency View capabilities. The CVV captures the variability with the help of variation points and dependencies. Each variation point in the view is associated with an artifact (i.e. feature tree, feature, architecture or a C header file) in SPL. Five types of variation points (i.e. optional, alternative, optional variant, variant and value) are specified in the view. These variation points include a description and information regarding its state (i.e. open or closed), the rationale behind binding, how the mechanism is to be realized.

As for the dependencies, three types of associations of variation points are specified in CVV. Predictable associations are used for the representation of variation points whose impact on the validity of the dependency can be determined prior to selection of the variants.

Directional associations are used for the representation of variation points, where dependency can only specify if the variant selection affects the validity of the dependency either positively or negatively but not definitely. Unknown associations represent variation points where the impact of variant selection on validity of the dependency is unknown.

Types of dependencies are distinguished to three as logical, numerical and nominal. Logical dependencies define a function valid providing the validity of dependency for the selection of variants of the associated variation points. Numerical dependencies specify a numerical value, which depends on the selection of the variants among the associated variation points. Nominal dependencies define a set of categories, where the binding of all variation points in association with dependencies map to one of each.

3.3.4 Bayesian Belief Network

As a feature diagram alone lacks the facility to help the designers select the best configuration of variants on the way to achieve the required QAs, Zhang et al. [16] proposed a Bayesian Belief Network (BBN) based approach to quality prediction and assessment for an SPL. Similar to Definition Hierarchy framework, BBN simply aims to represent and model the impact of functional variants (especially design decisions) explicitly on system QAs. These impacts originate from the interrelationships among variants (design decisions) and QAs. The examples of the interrelationships can be specified as the impact of one variant on many QAs, the impact of one QA on many variants, or impact of variants on each other in a competitive or synergetic fashion.

Impacts of the variants are represented as domain experts' knowledge and experiences derived from the development of similar projects on the same SPL. In addition to the analysis of relationship between the variants and the QAs, the BBN approach also deals with the uncertainity involved in the design decisions. The uncertainity is due to the risk that any configuration of the variants may lead to either high or low quality in terms of attributes and design decisions.

BBN model is structured around variables as QAs and design decisions which are represented as nodes in the model. These nodes are annotated with definitions. Directed edges in the tree model are used to relate a variant with one another (i.e. a design decision to a QA). Conditional probability is used for the quantification of the conceptual relationships between the variants or design decisions and QAs, value of which designates the domain expert's belief in how much a specific design decision has the impact on any given QA. The approach basically tries to enhance the comprehension of the impact of design decisions on QAs on the way to reveal more rational decisions.

3.3.5 Goal-oriented Approaches

Majority of software variability modeling methods are too design oriented, dealing too much with features and aiming relatively more on revealing architecture definition. The requirements are usually structured in definition hierarchies or feature models. This poses complexity during application as it necessitates a fairly large domain experience. During the requirement analysis process of SPL (i.e. analysis and selection process of variants), traditional SPL approaches pose problems when it is the case to deal with NFRs.

The goal-based model is introduced to the literature in order to solve these problems faced during the elicitation of NFRs. The model simply aims to model non-functional concerns explicitly and represent the intentionality of the system by relating the goals with traditional features. The goal is defined as the purpose of the system under development, having two types as hard goals and softgoals. Hardgoals are used for the satisfaction of functional features in a model whereas softgoals are used for the satisfaction of QAs required. The hardgoals and softgoals are dependent on each other in a way that the scope of variability in a model is defined by analysis of hardgoals but softgoals helps the designers to manage this variability in a most efficient way with exact definitions of quality criteria. [17]

3.3.5.1 NFR Framework

Mylopoulos et al. [18] proposed a goal-based visual variability analysis technique for explicitly modeling variants of different set of requirements for each product of a product family, in opposition to the case for only one product. Two sub-models are proposed as the functional goal model (dealing with goals and functional tasks) and the softgoal model (dealing with conditions and criteria that the system is to meet) derived from NFRs and QAs. Both of the sub-models are structured as AND/OR trees in which QAs are represented as soft-goals having their operationalizations included in the functional goal sub-model as tasks. Variability is represented through different OR paths of the functional goal model. Sub-models are related with each other through the correlation links defined by the NFR framework. [19]. The method offers to support the selection of a particular variant (represented in the functional goal sub-model) based on QA criteria, after the relations between two sub-models are established.

3.3.5.2 Feature Softgoal Interdependency Graph (F-SIG)

Many goal-oriented analysis techniques are based on QA analysis in a single system development. The goal-oriented analysis simply helps the designers to derive and represent QAs explicitly, specify the contributions from functional features to QAs and correlations among QAs, measure design decision impacts on QAs and finally selecting best design alternative satisfying certain QAs. On the other hand, being the extension of FODA, feature-oriented reuse method (FORM) is a well known domain analysis method to model common and variant requirements for PLs.

In Feature Softgoal Interdependency Graph (F-SIG) method, similar to goal-based visual variability analysis method [20], goal-based approach is aimed to be extended to a PL context by taking the advantage of QA analysis capability of goal-based approach. This approach is blended with PL domain analysis capability of FODA. The principal objective of the approach is to provide design model representing the interdependencies between variant features and QAs. In order to realize this objective, Softgoal Interdependency Graph (SIG) proposed by Chung et al. [19] is used to analyze NFRs by specifying softgoals and linking the interdependencies among variant features and softgoals. Similar to goal-based model, these linkages are established by means of correlation links defined in NFR framework. The approach is formed out to be F-SIG in the end.

Similar to some other QA variability modeling approaches such as COVAMOF and BBN, the F-SIG framework aims to aid the developers in assessing the impact of variant features on QAs required by the system. These impacts can be regarded as the restrictions to be defined on the behavioural requirements arising from the selected variant features. Against the difficulty in capturing the definitions and restrictions on QAs due to their vague nature, using softgoal concept in goal-oriented analysis assists the designers during feature modeling.

In the approach, interdependency is classified as structural and implicit. Structural interdependency is any explicit relation defined for features in the form of mandatory, optional, alternative and OR; for QAs in the form of only OR or AND all of which are imposed by modeling technique proposed by Czarnecki and Eisenecker [21]. Implicit interdependency is any correlation between features and QAs in the form of negative or positive as imposed by NFR framework proposed by Chung et al.[19].

F-SIG approach uses QA softgoal and claim softgoal types by discarding operationalizing goal, all of which are basic artifacts of SIG approach proposed by Chung et al.[19]. It is claimed that some features are regarded as they already include the role of operationalizing softgoals which are supposed to have contributions to QA softgoals.

The framework developed around F-SIG can help the designers in analyzing the interrelationships between design decisions and QAs. It does not include enough quantitative data to clearly address the best design decisions for any stated QAs in addition to a lack of tool support for the decision process.

3.3.5.3 Feature-Oriented NFR Analysis for SPL

Peng et al. [10] extended the goal-based models simply by materializing the NFRs in a way that NFR goals are linked with real-world context to aid realizing non-functional variability analysis over a whole domain.

The method elaborated in the context of the approach involves four major phases specified as following:

1. **Feature Context Construction:** As the starting point of all phases, a feature context model is constructed on an initially built functional feature model. Feature context construction is realized by modeling the real-world context from which non-functional concerns are derived in the form of execution scenarios, intents of human, events, social concepts etc.
2. **Non-functional Variability Identification:** Before identification of non-functional variability, goal presence analysis is realized. Some NFRs that are always desired by stakeholders are already included in almost every model, but context-specific NFRs has to be elicited with the help of NFR templates as heuristics for the goal-presence analysis.

   Being the product of goal presence analysis, non-functional goals are further analyzed into sub-goals with different levels. Against one of the primal handicaps of other QA variability modeling approaches (including goal-oriented approaches), the method takes a chance to propose optionality at PL level by specifying three levels for each NFR goal as low, medium and high. Higher level NFRs always replace lower ones. But if an higher level NFR conflicts with another NFR, the lower levels are to be reserved as well. Concept of conflicts between NFRs on the level of operationalizations is a significant part of the variability analysis, since non-functional variations arise from tradeoffs of these conflicting NFRs.

   For each of these NFR levels, operationalizations are further specified to set up relations in the form of conflicts and dependencies. Besides, the operationalizations are further evaluated for all NFR levels considering the possibilities of their satisfiability such that if an operationalization is evaluated to be unsatisfiable, then it should be removed. Besides the operationalizations define the presence of NFR levels or their relations such that; if all operationalizations of an NFR level are removed, then it should be removed or if all operationalizations of an NFR level are partially satisfiable, then it should be adjusted to be optional.

During integration of operationalizations into conventional feature models, as of the bridges between the NFR levels and functional features, there emerges the need for definition for two types of operationalizations, namely dynamic operationalizations that can be regarded as a sub-feature of the affected functional feature and static operationalizations that can be regarded as restrictions on the affected functional feature. Besides, for the instances of dynamic operationalizations affecting multiple functional features are defined as crosscutting features, interactions with functional features of which are required to be recorded.

3. **NFR Integration:** NFR-related operationalizations are involved/incorporated into the feature model in parallel to modification on functional features based on the relations with non-functional concerns.

4. **NFR-oriented Decision Modeling:** Finally, a feature decision tree model with both functional and non-functional concepts is built where all NFR conflicts, NFR dependencies, environmental dependencies and constraints are taken into account. In decision model, only variability-related features are involved.

The structure of the feature model produced by the approach is constructed on the variation points of optional, alternative and OR features. Optionality as the basics of variability is utilized differently in the approach of Peng et al. such that; in conventional feature models, an optional functional feature is either included in the model or not, whereas an NFR can never be excluded from the model. It is intrinsically included in the model and it has to be represented explicitly for an effective management of QAs. That is why the different levels for soft goals are defined in their model.

Concept of cardinality is also involved in the model due to the need of representation of numerous variants that can be chosen for a single variation point of OR type. Cardinality is represented as [minimum number of variants … maximum number of variants] in the feature model.

As an important intermediate product of the modeling process proposed by the approach, NFR graphs are critical such that four important elements of the whole process are encapsulated in. These are NFR goals with sub-goals, identified NFR levels for each respective NFR sub-goals, suggested operationalizations for the implementation of each NFR levels and required environmental conditions for specification of operationalizations.

Four different types of relations between these elements are defined in the concept of NFR graphs, specified as, (1) AND/OR decomposition of a goal to its corresponding sub-goals, (2) relation between NFR goals and its corresponding NFR levels, (3) AND/OR decomposition between NFR levels and its corresponding operationalizations and (4) environmental dependencies between operationalizations and environmental conditions (for the evaluation of applicability of each operationalization). As an illustration of notations included in an NFR graph, a sample NFR graph of Security and Usability modeled with reference to Computer Aided Grading System (CAGS) domain is demonstrated in Figure 3.3 below.

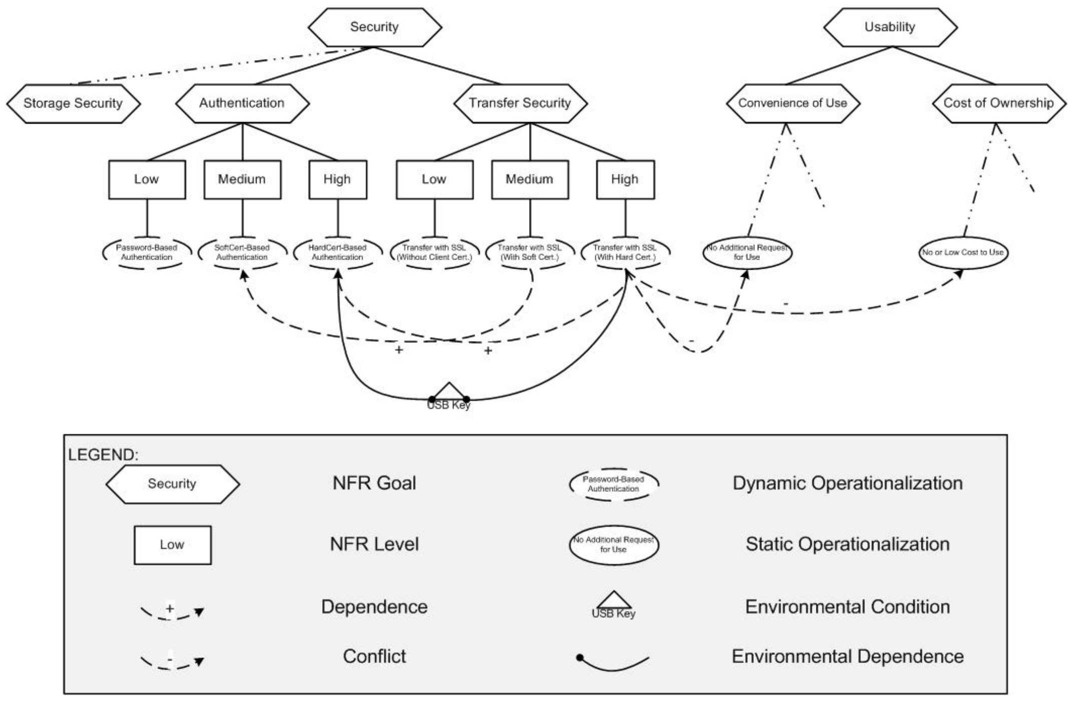

Security
Usability
Storage Security
Authentication
Transfer Security
Convenience of Use
Cost of Ownership
Low
Medium
High
Low
Medium
High
Password-Based Authentication
SoftCert-Based Authentication
HardCert-Based Authentication
Transfer with SSL (Without Client Cert.)
Transfer with SSL (With Soft Cert.)
Transfer with SSL (With Hard Cert.)
No Additional Request for Use
No or Low Cost to Use
+
+
-
-
USB Key
LEGEND:
Security
NFR Goal
Password-Based Authentication
Dynamic Operationalization
Low
NFR Level
No Additional Request for Use
Static Operationalization
+
Dependence
USB Key
Environmental Condition
-
Conflict
Environmental Dependence

Figure 3.3: Sample NFR Graph for NFR Goals in CAGS Domain (adapted from [10])

In terms of classification of interrelations amongst artifacts of NFR integrated feature model, as illustrated in the Figure 3.3 above, structural interdependencies can be realized in the form of AND/OR decomposition of goals to its sub-goals and decomposition of NFR levels to its operationalizations.

The interplay between NFR sub-goals and their corresponding NFR levels can be modeled in the form of either structural or implicit interdependencies since any level of NFRs mutually excludes the other levels. Defining conflicts relationship as an implicit interdependency for each of NFR levels belonging to the same sub-goal has the same affect on variability with defining alternative relationship as a structural interdependency between a sub-feature and its corresponding NFR levels. From the variability point of view, they are considered to have the same effect on the variability of the model. It would be better to use structural rather than implicit interdependency for the sake of ease of perception and understandability of the model.

Other instances of implicit interdependencies can be realized between NFR levels and functional features in the form of relations between functional features and either of static or dynamic operationalizations or crosscutting features. Besides, there may exist relations amongst operationalizations no matter to which NFR level they are belonged to. These relations are in the form of conflicts or dependences (that can be regarded as excludes or requires relationships respectively). One other implicit interdependency is defined as Environmental Dependence specifying the relation between environmental condition (i.e. required hardware or medium due to selected functional feature) and the corresponding operationalization.

The concept of dependency among NFR goals, functional features and environmental conditions is beneficial for the integration of non-functional perspectives into the decision model.

3.3.6 Tool Support for Non-functional Feature Modeling

There is no generally accepted modeling tool for QA variability. COVAMOF, Definition Hierarchy, EFM and Goal-based Models provide their own tools that have been developed specifically for each design implementation; BBN uses a commercial tool named Hugin for the automation of inferences and F-SIG uses MS Visio for the representation of the models.[22]

3.4. Discussion of the Literature

In order to take the variability derived from NFRs under control, a detailed study has to be provided at very beginning of DE particularly in Domain Analysis phase. As individual products are under development, this study has to be supported and extended with the experiences gained during Product Requirement Analysis phase of AE by sustaining a two way feedback between respective sub-processes of DE and AE.

But it is not that easy to represent NFRs clearly for their ease of perception during the analysis of the domain and the development of applications. Besides, the efforts and approaches fall short in providing an integrated feature modeling framework to capture NFRs together with FRs.

As the outcomes of product's NFRs analysis can be stated as an important input for the definition of product features; prior to design and configuration of the product is completed, a consistent feature analysis has to be realized with proper methods helping for elicitation of NFRs. Introducing NFRs to the feature model ontology is expected to enrich the feature model ontology with description of relations, constraints, restrictions among QAs and functional features.

In the scope of SPL, variability is defined with the help of variation points. These variation points can define decision points in conjunction with their possible alternatives in the form of functional or non-functional (quality) aspects. These decision points can form the basis of NFR-integrated variability models which can provide comprehensive variability decision support for AE with both functional and non-functional considerations. Right after the structural and implicit interdependencies of any feature model are defined and model validation is performed (i.e. the feature model is valid if at least one product can be derived from it), FAMA can guide the designers towards a valid and complete feature selection (decision model support with the help of the rules of relations defined among the QAs and functional features) which would help in making evaluation of the integrated feature model basically in terms of its variability.

During feature modeling, the control over these variation points is crucial since identifying and understanding the dynamic semantics of systems is a requisite for product design activities. Most of the time these points derived from NFRs are not formally identified in a feature analysis with details and there exists no competent mechanism to capture them explicitly. In order to have such applicable feature models, the variation points derived from the analysis performed for NFRs are to be explicitly identified within a feature model and are to be integrated suitably with functional features. This is essential for Domain Architecture Design phase of DE in extension with Product Design Analysis phase of AE.

The tendency to use feature diagrams in feature models as the basic aid for modeling proves its competence as variability representation technique, not only for the analysis of FRs but also for NFRs, since its usage is adopted by most of the recent approaches.

During the literature review, state of art QA variability modeling approaches have been assessed with respect to their potential contribution to variability model development, taking their capability to reveal the interplay of QAs with functional features into account.

As of being one of the QA modeling approaches, Definition Hierarchy concentrates on analysis and classification of requirements which practices especially on the problem domain leaving little effort for design activities in the solution domain. Even design decisions are realized in parallel to the elicitation of the requirements. There has to exist a balance of endeavor between these two domains.

During variability modeling of QAs, COVAMOF emphasizes dependency analysis between variation points. However, in order to determine the applicability of design decisions on variation points, some restrictions or conditions such as environmental dependencies on operationalizations of the variants have to be taken into account.

As the impact of the variants are specified based on domain experts' belief regarding the knowledge and experience derived from the development of similar projects on the same SPL, BBN lacks objectivity and correctness for the instances of immature SPLs with on-going development on their competence.

QA variability modeling approaches mentioned above remain short of responding to the need of different levels of QA optionality at PL level except the approach proposed by Peng et al.[10]. This need for modeling qualitative QA knowledge in feature domains can be expressed with an example of cost requirement. Concerning all product candidates to be derived from the same SPL domain, low cost attribute can have a high level of priority for one family member whereas it may have low priority in another. In addition to advantage of availability of optionality at PL level, goal-based approach serves the opportunity to make trade-off analysis between different QAs for the sake of satisficing different conflicting NFRs with the help of operationalization analysis.

Besides, most of the QA variability modeling approaches mentioned so far fall short of responding to the need of modeling QAs with a quantitative analysis, except EFM approach proposed by Benavides et al. [13]. In the model, functional features are decorated with attributes as characteristics of a feature that can be measured such as availability, cost, latency, bandwidth etc. Furthermore, relations between the attributes attached to several functional features or non-functional features can be constructed.

Among the others, EFM approach of Benavides et al. [13] and Feature-Oriented NFR Analysis method of Peng et al. [10] seem to complement each other for a better variability analysis of non-functional features in terms of their facilitations for quantitative and qualitative quality aspects respectively. Inspiring from what Bartholdt et al. [23] propose in their study, qualitative and quantitative QAs can be integrated to functional features in the same model. By establishing interrelations between the artifacts of Feature-Oriented NFR Model and quantitative QAs attached to functional features (inspiring from EFM approach), the integration of functional features with non-functional features can be achieved on the same model. Utilization of FAMA tool would help to deduce evaluations on the integrated feature models in terms of its variability.

The basic and most important contribution of this QA integrated feature modeling approach is that it gives domain designers the chance to elicit and expose the quality knowledge hidden in the specifications of the requirements. Moreover, it allows the product designers to manage these systematically elicited and modeled quality knowledge in terms of its variability. In other words, the approach serves the ability to control variants originating from qualitative and quantitative quality aspects.

In the next chapter, an approach inspired from a synthesis of the aforementioned quantitative and qualitative non-functional feature modeling approaches is implemented on the same feature model, in accordance with the considerations presented in this section.

CHAPTER 4

FEATURE MODELING OF METEOROLOGICAL MEASUREMENT AND LOGGING SYSTEM DOMAIN

4.1. Introduction

In this chapter, with the aim of providing a proof of concept and detailed illustration for feature modeling using the proposed Complementary Quality Modeling Approach first, the standard feature model of the Meteorological Measurement and Logging System (MMLS) domain, will be introduced together with its SPL-oriented specifications for the modeling of the system.

Subsequently, the reasons why this sample domain is selected for feature modeling will be provided in addition to the discussions regarding the experience acquired during the design.

Afterwards, before all else, MMLS domain will be modeled in a conventional feature modeling approach wherein non-functional features and quality concerns are not peculiarly taken into account during variability modeling.

Towards the end of the chapter the conventional feature model developed priorly is extended for the sake of supplementing variability modeling with QA concerns. Variability modeling specifications, implementation details and experiences obtained during the development of the model extended with QA aspects are provided as well.

4.2. Meteorological Measurement and Logging System (MMLS)

As the primal objective of this study, the elicitation of variability knowledge inherent in a domain is a prerequisite for the design of domain architecture and implementation of the domain with suitable artifacts. For the purpose of efficient exercise on variability concerns during development of a feature model, a reference domain that is easily adaptable to SPL-oriented DE and adequate in terms of diversifiability potential is a requisite.

Practicing on variability modeling of MMLS domain is preferred as it conforms to the considerations outlined above. Based on its system specifications, basically, MMLS aims to measure, log and display meteorological information necessary for weather logging and forecasting applications. For this purpose, the system embodies different types of sensors each of which are specialized on the type of the information to be collected from the environment in various weather conditions. Different types of sensors come along with different types of interface specifications as they are procured of commercial-off-the shelf (COTS) hardwares. As the system is assigned to exploit logging and displaying facilities as well, a software configuration is required to collect the data transmitted from several sensors with different electrical interface specifications in order to convert this information for processing purpose followed by logging and displaying functionalities. The system is required to offer different software configurations as it needs to be compatible to work with different types of display units which are to be procured as COTS hardwares similar to the case in sensor units. Besides, the system needs to be configurable in order to transmit different types of processed information to external systems, interface specification of which is dependent on the system that MMLS is designed to be integrated with.

In the light of the basic system specifications summarized above, in the scope of this study, the software to achieve aforementioned tasks is proposed to be a single software configuration unit which is composed of different software components.

As the main purpose of the study, the variability knowledge to be elicited from system requirements will be managed inside the internal structure of these components in terms of different parts. These parts are some kind of problem domain counterparts of modeling artifacts in solution domain, namely the features. Based on feature decisions provided during AE phase, these parts are assembled to form the desired configuration and constitute the required components.

Besides, the study is concentrated solely on the dynamics of feature modeling aspect of the reference domain of MMLS. The possible relations and interactions with other phases of SPLE will be provided with discussions as well.

In the present work, the Feature Model together with its extension to quality modeling aspects of MMLS domain is developed by the author to be used as the proof of concept and revised by experienced engineers from Aselsan Inc. for its consistency. Besides, during development of MMLS domain for SPL, experience formerly gained in course of development of an MMLS software configuration delivered to customer and built in a layered architecture within a non-SPL context is utilized as guidance and inspiration. Similar set of requirements are used as the driving force for the specification of features for software configuration needed by all stakeholders (i.e. domain experts, application engineers and customer).

4.3. Feature Modeling of MMLS Domain

Prior to the construction of feature model of MMLS domain, fundamental tasks to be achieved are needed to be outlined with the help of a comprehensive elicitation of domain requirements. These tasks are to be achieved by any probable software configuration produced during AE phase. Following the definition of these fundamental tasks, different kinds of features are to be derived and augmented on the basis of their respective root features where the emphasis is laid especially on the variability concern.

In the light of these, any system produced from MMLS domain has to interact with users in order to realize user-dependent functionalities. This brings the need of Graphical User Interface composed of a number of components acting like a bridge between the user and other components of the software assigned to realize different functionalities of the system.

Besides, as can be deduced from the name of the domain, one of the main duties to be achieved by any instance of MMLS domain is to realize the logging facilities where a number of components control the communication between the sources of data to be stored and the database acting as the main storage unit. Additionally these components have to provide services to the Peripheral Retrieval Components in order to realize the back-up facilities for the protection of information on the database. Moreover, some of the components that can be classified in this group have to realize some further maintenance tasks in order to ensure the availability of the system.

The system needs to communicate with different types of sensors and display units to collect and represent the measured data respectively. Therefore, the tasks to be achieved by the system hardware interface has to be assigned to some components that can be grouped under the definition of Peripheral Retrieval Components, as referenced in Figure 4.1 below. Furthermore, these components have to achieve interaction with a printer as it is to be included in any product configuration.

Communication between these three types of component groups is to be controlled and organized by System Management Components.

This classification of components eases the perception of the distribution of tasks amongst the components of MMLS domain regarding their fundamental roles outlined above.

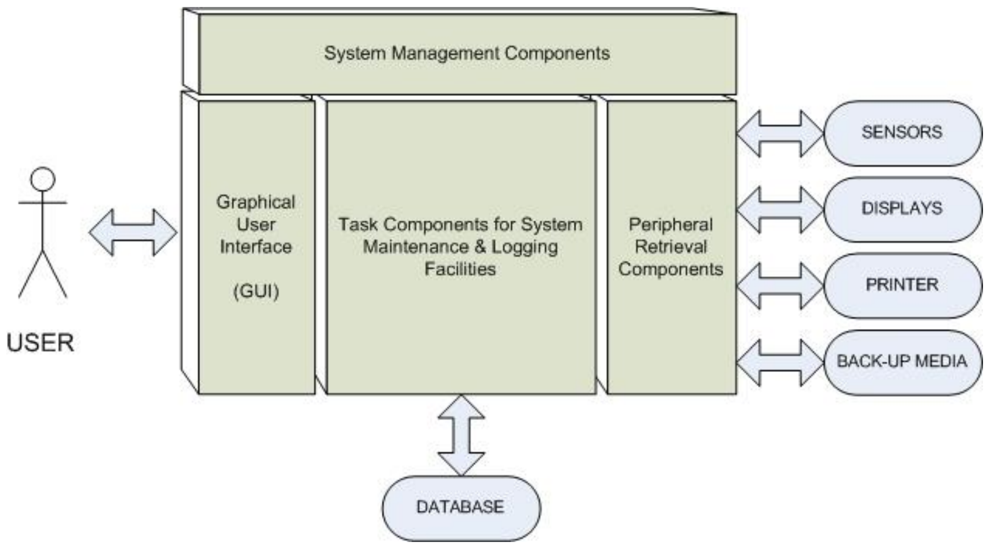

System Management Components
Graphical User Interface (GUI)
Task Components for System Maintenance & Logging Facilities
Peripheral Retrieval Components
SENSORS
DISPLAYS
PRINTER
BACK-UP MEDIA
USER
DATABASE

Figure 4.1: Basic MMLS System Architecture

4.3.1. Feature Context Analysis

As a starting point for the definition of the scope and context of MMLS feature model, the classes of component groups with reference to Figure-4.1 have to be utilized. These groups are some kind of a classified compilation of solutions developed from the requirements that any instance of MMLS domain has to meet. For the sake of meeting these requirements, varieties of features can be defined and extended on the grounds of each of these basic classes.

It must be noted that, during the construction of feature diagrams, all the information derived from the expected capabilities of the system is not required to be projected on the feature model. This is due to the fact that, basic need for the construction of a feature model is to manage the variability information included in a domain. Redundant usage of common features withinside a feature model degrades the quality of a feature model considering its power to exploit variability knowledge. Consequently, in the scope of this work, as introduced in the following sections, variant features of the domain are included in the feature model, in which the commonality knowledge is modeled inherently and indirectly (discussions on proof of concepts are to be provided in Chapter 5).

4.3.2. Feature Variability Identification

In the discussions hereunder, the textual explanations on the feature model structure are provided in addition to their representations on diagrams for the sake of clarification.

As depicted in the feature diagram in Figure-4.2, the uppermost level of MMLS Feature Diagram consists of six variation points, each of which are composed of a group of features.

Some of these features are variation points as well, in which at least two features are present (to be specified afterwards); some of them are the variants (a.k.a the leaf nodes of the feature tree) to be included directly in a product configuration.

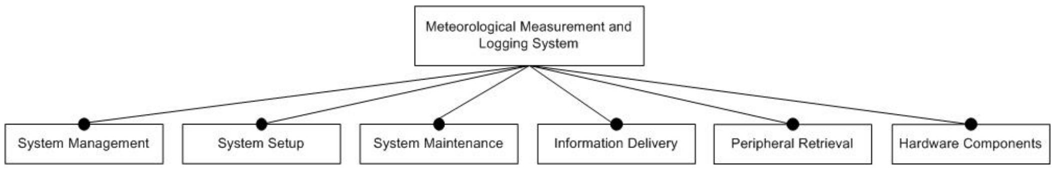


Figure 4.2: Feature Diagram for MMLS Domain – Uppermost Level

As demonstrated in the diagram, all the variation points in the uppermost level are modeled as mandatory feature groups, such that any instance of MMLS product configurations has to possess System Management, System Setup, System Maintenance, Information Delivery, Peripheral Retrieval and Hardware Component facilities with any combinations of their respective variants. In consistency with the definition of basic MMLS System Architecture provided in *Section 4.3.1 Feature Context Analysis*; these feature groups substantially reflect the content of their counterparts in the solution domain as classified groups of components.

As mentioned in *Section 2.3.2 Feature Modeling* and referenced in *Figure 3.1*, interrelationships among the functional features are classified as structural and implicit interdependencies. The rest of the feature model detailed below the uppermost level is basically composed of these two types of feature relationships. For the ease of their representation, the two classes of interdependencies are handled separately in the following sections with their constitution rationale provided.

4.3.2.1. Structural Interdependencies

In this section, MMLS feature model is examined in terms of its structural relationships in between feature groups and their respective sub-features. As referenced in *Section 2.3.2 Feature Modeling*, four types of relations are utilized in the model, namely mandatory, optional, alternative and OR relations. It should be noted that, unless otherwise specified, all OR-relations implies a selection of at least one feature rather than none out of all variants in the feature group hold by the parent node.

System Management is the principal feature group of the model. Literally, upper layer functionalities; management of security, maintenance, logging facilities; operational management of user interface and hardware interfaces in addition to sustaining the communication in between all these features are to be handled within this group of functions. For instance, the process in which a measured data is collected, delivered to processor following with transmission to external systems or displayers through hardware interfaces is managed by this group of functions. In addition to these facilities, basic functions such as ‘Initialize System’ and ‘Shut-down System’ are not included in the model since they are treated as commonality points due to their non-contributory nature with respect to variability concern.

Apart from these functional capabilities, as demonstrated in *Figure 4.3,* three variation points, namely, Authentication, System Log Management and Failure Management are incorporated as sub-feature groups into the feature model together with their variant contributions. Authentication feature group is modeled as optional whereas System Log Management and Failure Management feature groups are modeled as mandatory. The rationale behind modeling Authentication feature group as optional is to leave the decision of possessing a system with extra security precautions to the user. Authentication serves two alternatives as Hardware-based and Password-based Authentication facilities meaning that no such system configuration exists which supports both types of authentication. HW-based Authentication is to be handled by one of the most conventional form of its utilization such that as long as this function is selected, a USB Security Key is needed to initialize the system. PW-based Authentication serves two optional functions: a reminder that notifies the user periodically to change the system authentication PW and a facility to save a master-key attached to the standard authentication password which assures a two-fold security for authentication.
System Log Management feature group holds two optional features with OR-relation: history track facilities for System Setup Information and Operational Information (i.e. log-in – log-out information, changes in the status of system SW, measured data exchanges between sensors and external systems).

Finally, Failure Management feature group holds two levels of failure management facilities: Unit-level and Module-level. In Unit-level Failure Management, a low level of fault detection is utilized such that status knowledge of replaceable units (i.e. sensors) are monitored with the help of data communicated through the interfaces. Module-layer Failure Management is a higher level of fault detection mechanism that monitors failures possible to emerge between the HW-components in terms of their inter-accessibility. As

Module-layer Failure Management mechanism encompasses all functionalities of Unit-Level Failure Management, these two options are modeled as alternative to each other in the feature model.

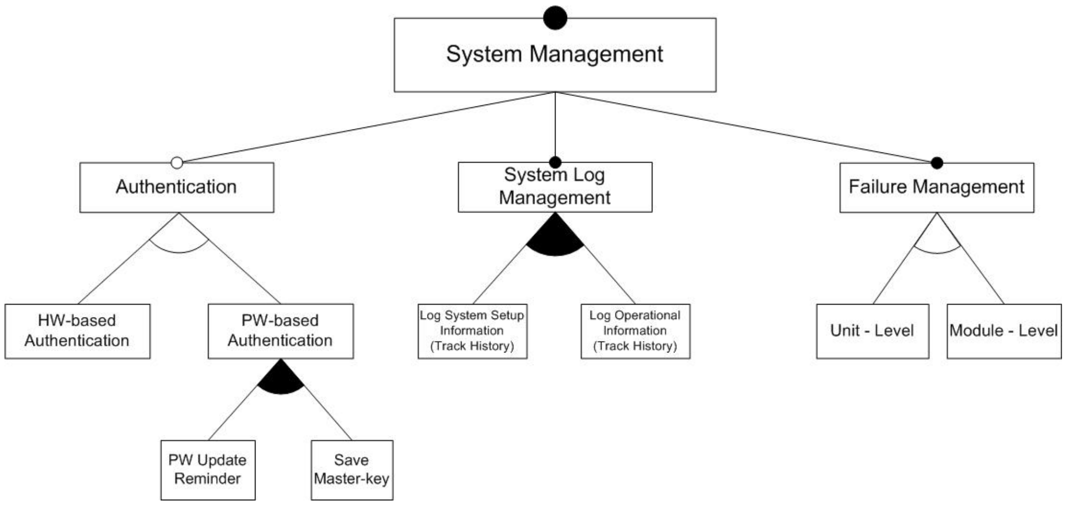


Figure 4.3: Feature Diagram for System Management

System Setup is another uppermost-level feature group included in the feature model. Similar to the case in System Management feature group, the functions incorporated under this group are filtered out in terms of their contribution to variability of the overall feature model. Primary mission that could be classified under this group is the realization of any type of system setup functionalities such as setting the type of transmission channel or setting the time and date of the operating system. For this reason, they are not incorporated into the feature model considering their lack of contribution to variability concern.

As depicted in *Figure 4.4*, System Setup feature group offers four variation points namely Set/Change Data Recording Frequency, Set/Change Data Transmission Frequency, Set Resolution and Set Sensor Parameters which are interrelated through OR-relationship with each other. Set Sensor Parameters feature offers the capability of setting the internal parameters (i.e. initialization and offset values) of some specific sensor types. Set/Change Data Recording Frequency and

Set/Change Data Transmission Frequency features provide the ability to adjust the frequency levels of transmission to data storage and display units respectively.

Any product instance of MMLS domain is in compliance with 800x600 resolution. As for Set Resolution feature, system offers to switch from this default resolution to any of the selected resolution alternatives. Depending on the type of display units included in system configuration (i.e. this feature is meaningful as long as Monitor Keyboard Drawer is included in the HW-configuration - to be detailed later in *Section 4.3.2.2 Implicit Interdependencies*), system offers at least one of any two resolution alternatives: 1280x1024 and 1024x768. These features are modeled through OR-relation with each other, since Set Resolution feature serves the ability to switch between any resolution scales included in product configuration.

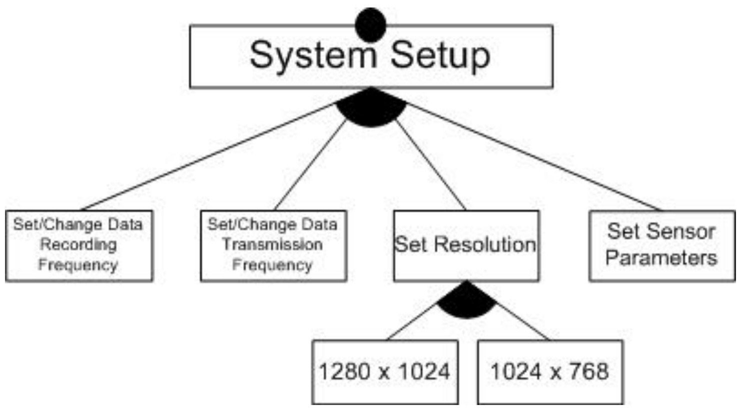

System Setup
Set/Change Data Recording Frequency
Set/Change Data Transmission Frequency
Set Resolution
Set Sensor Parameters
1280 x 1024
1024 x 768

Figure 4.4: Feature Diagram for System Setup

As a precaution against probable failures to emerge during the normal operation of the system, feature model is decorated with a number of variants classified under different feature groups with a System Maintenance parent feature on top, as portrayed in *Figure 4.5*. System Maintenance includes three feature groups, by name, System Test, Database Maintenance and Store/Restore Database Back-up.

System Test feature group includes three types of tests related by OR-relation with their parent node, namely, Start-up Test, On-line Test and Off-line Test. Basically, Start-up test is performed for checking mission readiness, On-line Test ascertains the persistence of system performance, Off-line test acts as the inspector to specify the defective component in cases of system failure.

In relation with Failure Management Levels (to be clarified in *Section 4.3.2.2 Implicit Interdependencies*), main capabilities of these testing types are mentioned as follows:

Start-up Test offers the following facilities:

- Controls and reports sensor status
- Initiates as soon as the system is powered
- Can be stopped and bypassed at any time

On-line Test offers the following facilities:

- Detects the existence of a unit-level fault, is more efficient if followed by Off-line Test in order to specify the specific location of a fault in an HW component (for instance On-line test detects a failure related with the transmission from any specific sensor or printer, but it is the off-line test that detects the failure is whether caused by the module which processes the signal or the source is defective)
- Controls and reports status of communication with sensors or other basic HW components (i.e. data storage unit) such that the test detects a failure if only no signal is received from the signal source
- Is continuous, initiates with start-up and runs until system shut-down without the need of operator intervention and can not be stopped or bypassed

Off-line Test offers the following facilities:

- Performs more comprehensive test on different HW modules (i.e. PCBs, processors etc.) with the help of fault diagnosis algorithms, in addition to unit-level facilities provided by On-line Test
- Controls and reports operating status of supply voltages, processor and database and data transfer lines in addition to units (i.e. sensors or printer)
- Executes with the initiative of the user

Any failure in database is not desired in order to avoid sudden loss of information especially for the cases that backed-up version of it is not available. In order to realize these preventive facilities, following operational capabilities are modeled under the Database Maintenance feature group with OR-relation among each other; Database Clean-up clears all the data stored in the database for a fresh start to logging (especially beginning time of logging is of concern), Store/Restore Back-up Database to store/restore data to/from any type of back-up media in order to either observe previously loaded data or continue to logging over and transfer the data contained in the database to selected type of back-up medium. As part of Store/Restore Back-up Database feature group, three types of storage media is available for the purpose of backing-up the information contained in database so as to prevent data loss due to any failure in database. These storage media types, namely USB, DVD and CD are modeled as an alternative to one another such that none of them is permitted to co-exist with the other due to its redundant usage.

System logging works in accordance with First In First Out (FIFO) rationale. As soon as the system reaches its maximum logging capacity (as restricted by means of the data recording hardware), most recent data is overwritten on the latest data. Keeping this in mind, if preventing the system from loss of critical recorded data is of concern, Database Clean-up and Store/Restore Database Back-up features are requisite for the desired product configuration. Apart from these, System SW Update is served as optional function for any case of failure in the system SW.

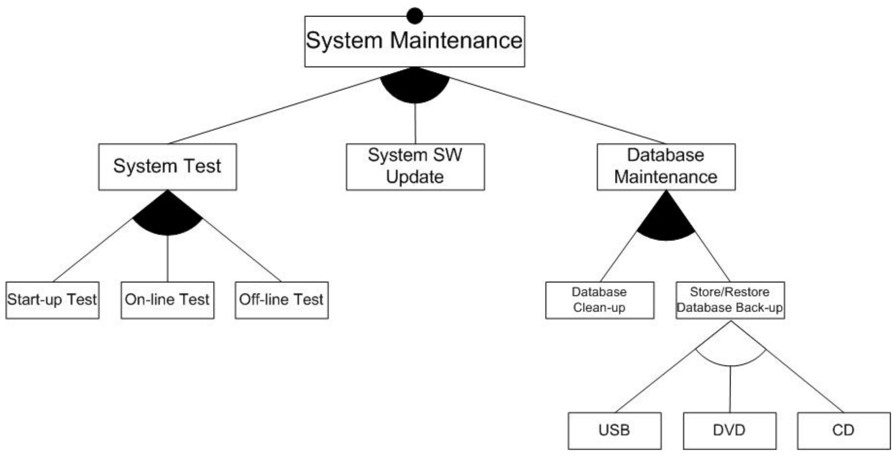


Figure 4.5: Feature Diagram for System Maintenance

Information Delivery feature group consists of functionalities that are particularly related with the forms of information transmission to user, most common example of which is to provide visual information by means of any selected visual fashions. Based on the variety of these visual resources, features are classified under the variation points of Graphical User Interface (GUI) and information delivery on printed material. These two facilities are indispensable elements of any MMLS domain, therefore they are modeled as mandatory variation points. By means of GUI, delivery of statistical information through graphics (i.e. charts, measured value vs. time plots), maintenance screen as for monitoring detailed system status and data transmission screen for observing the status of delivered or received messages (i.e. measured sensor data) through the interfaces is served as variants. Owing to the presence of the printer in the configuration, options of print-out capabilities for either textual statistics or graphical statistics are available as variants.

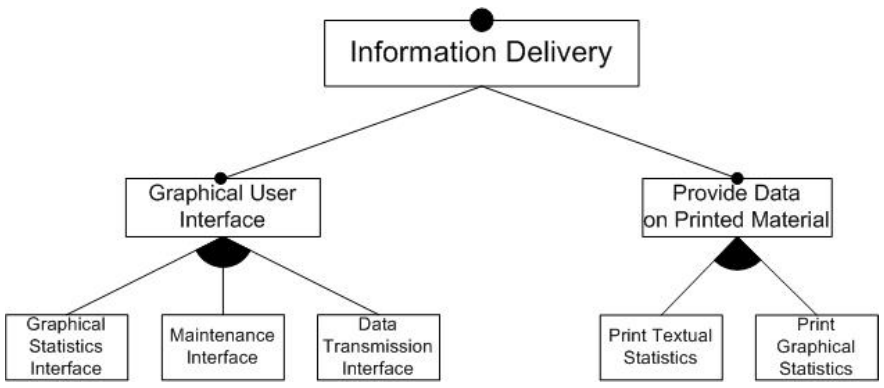

Information Delivery
Graphical User Interface
Provide Data on Printed Material
Graphical Statistics Interface
Maintenance Interface
Data Transmission Interface
Print Textual Statistics
Print Graphical Statistics

Figure 4.6: Feature Diagram for Information Delivery

Peripheral Retrieval is another feature group that holds the variants of hardware interface types to ensure the communication not only within the sub-units of the MMLS but also with external systems. As of features to be classified under Peripheral Retrieval feature group but not to be reflected on the feature model due to commonality; any instance of MMLS domain provides the capability to let the user edit the settings for its interface and realize overall communication both inside and outside its borders.

System communicates with the aid of either RS422, RS232 or VGA interfaces leading to mandatory relation with its parent node. To be detailed later in *Section 4.3.2.2 Implicit Interdependencies*, types of sensors and display units included in the configuration implicitly specifies the presence of either of these three types of interfaces. Printing Interfaces offered are available for a wide spectrum of printer types which contributes the system a high standard of modularity and integrability with COTS printer products. In order to broaden this spectrum, MMLS feature model is designed to support integrability with former versions of printing interface types (i.e. Serial Port and Parallel Port) but also contemporary versions (i.e. USB Port and Ethernet/LAN). It must be noted that these four types of printing interfaces are connected to their parent feature through alternative-relation since only one type of printer interface per product configuration is supported by the MMLS domain.

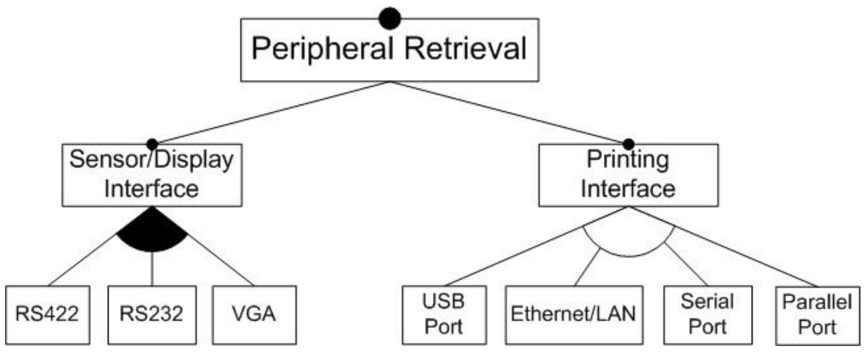


Figure 4.7: Feature Diagram for Peripheral Retrieval

Different from the feature groups mentioned so far, Hardware Components feature group deals with critical hardware elements of the MMLS domain. From this point of view, capabilities available in this group complement the design decisions realized during the development of the software components.

Being one of the most critical entities of the MMLS domain, Hardware Components feature group poses its significance when the limitations regarding the realization of any product configuration is of concern. Decisions based on the user-defined system requirements directly effect the hardware component configuration in the first place. Subsequently, specification of hardware component configuration puts borderlines on the varied capabilities of the software components. This is due to the fact that most of the features in this group possess specific characteristics on their own which causes potential conflicts with a number of varied features of any software configuration. This situation is handled by implicit interdependencies that are established between SW and HW features as they are detailed in *Section 4.3.2.2. Implicit Interdependencies*.

Measurement of weather conditions, logging the measured or processed data and representation of the transformed information are the principal duties of any product derived from MMLS domain. Therefore, Sensors, Displays, Printer and Data Storage Units feature groups are modeled as mandatory variation points in the sub-tree of Hardware Components. It should be noted that any product configuration derived from MMLS domain owns basic level of GUI on a built-in display unit that has the capability to meet basic and common demonstration requirements of the system. Due to their widespread usage for weather condition monitoring, Wind Speed and Direction Sensor, Ambient Humidity and Temperature Sensor and Air Pressure Sensor are incorporated in the model through an OR-relation with their parent node. Displays feature group offers two kinds of display units: Monitor Keyboard Drawer is the more sophisticated one with PC-like capabilities such as integrated keyboard and touchpad in addition to high resolution display in varied resolutions; LCD Display is the moderate one with average display quality and less user-friendliness. Since common visual representation requirements of any MMLS product configuration are already fed by built-in display that is already included in any product configuration, additional varied user-requirements are to be satisfied by any one of these two types of display units which are modeled as alternative to each other. Printer feature group includes two varieties of printers as Inkjet and Laser types.

So as to realize recording all manner of information (i.e. measured data, setup data etc.) produced by the system, three different types of data storage units are modeled through the alternative relation with their parent nodes. Type A and Type B units are Solid State Disks (SSD). They offer lower levels of storage capacity with higher cost and performance with respect to Type C which is a standard Hard Disk Drive (HDD). The specifications of these three types of data storage units are detailed in *Section 4.4 Integrating Complementary Quality Modeling with Functional Feature Model of MMLS Domain*. These specifications are used as basic modeling commodities of quality modeling aspects.

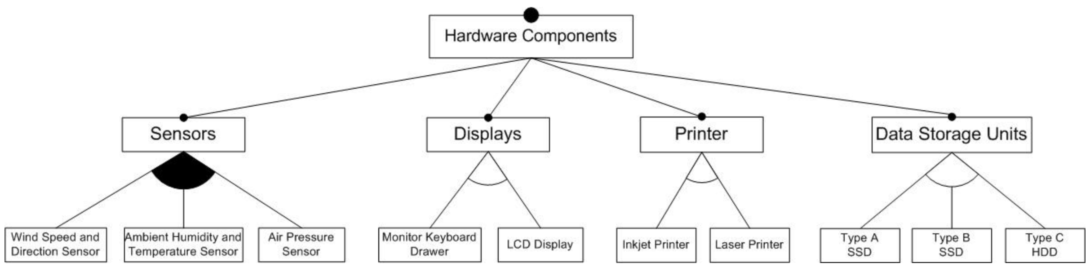


Figure 4.8: Feature Diagram for Hardware Components

4.3.2.2. Implicit Interdependencies

In order to ensure a consistency in terms of interrelations among the features of MMLS domain, construction of implicit interdependencies on the basis of structural interdependencies is indispensible. In this section, MMLS feature model is discussed in detail by revealing the rationale behind its constructed relationships. It must be noted that two basic types of relationships are practised in the model, namely requires and excludes relationships. Requires is a directional relationship between the features such that a requires relationship defined from feature A to B means feature B is required to be included in the product configuration if A is included. Different from the requires relationship, excludes relationship is bi-directional meaning that presence of any of feature pairs interrelated with exclude relationship can not co-exist in the same product configuration.

There exists numerous implicit interdependencies between the features of MMLS domain. Since it is cumbersome to depict all these relations on feature diagrams, as is the case in structural ones, textual descriptions regarding each and every one are provided in this section.

For the sake of understandability of textual descriptions to express the rationale behind relationships between these features, they are indicated by connecting two feature (source feature and target feature) specifications with requires and excludes expressions. For instance, System SW Update requires Maintenance

Interface means that if System SW Update capability is included in the product configuration, than Maintenance Interface facility is required to be included in the product configuration.

These feature relationship expressions are compiled in Table 4.1 below to assist the textual descriptions provided hereunder.

Table 4.1: Implicit Interdependencies of MMLS Model

ID	Source Feature	Dependency Relationship	Target Feature
1.a	Unit-Level	Excludes	Off-line Test
1.b	Module-Level	Requires	Off-line Test
1.c	Off-line Test	Requires	Module-Level
2.a	Set/Change Data Transmission Frequency	Requires	Data Transmission Interface
2.b	Data Transmission Interface	Requires	Set/Change Data Transmission Frequency
3.a	Set Resolution	Requires	Monitor Keyboard Drawer
3.b	Monitor Keyboard Drawer	Requires	Set Resolution
4.a	Off-line Test	Requires	Maintenance Interface
4.b	System SW Update	Requires	Maintenance Interface
4.c	Database Maintenance	Requires	Maintenance Interface
5.a	Wind Speed and Direction Sensor	Requires	Set Sensor Parameters
5.b	Set Sensor Parameters	Requires	Wind Speed and Direction Sensor
5.c	Wind Speed and Direction Sensor	Requires	Off-line Test
6.a	Inkjet Printer	Excludes	USB Port
6.b	Inkjet Printer	Excludes	Ethernet/LAN
6.c	Laser Printer	Excludes	Serial Port
6.d	Laser Printer	Excludes	Parallel Port
7.a	Wind Speed and Direction Sensor	Requires	RS232
7.b	RS232	Requires	Wind Speed and Direction Sensor
7.c	Ambient Humidity and Temperature Sensor	Requires	RS422
7.d	Air Pressure Sensor	Requires	RS422
8.a	LCD Display	Requires	RS422
8.b	Monitor Keyboard Drawer	Requires	VGA
8.c	VGA	Requires	Monitor Keyboard Drawer

- Rationale for 1.a, 1.b & 1.c;

  Unit-Level failure management capability is the core of On-line test feature. But as mentioned earlier, Module-Level failure management is the enhanced version of Unit-Level and is an alternative to it. Off-line Test holds Module-Level failure management as its essential utilization. Therefore, there should exist no configuration that includes Unit-Level Failure Management capability along with Off-line Test. Besides, Module-Level failure management capability and Off-line Test feature mutually require each other. It should be noted that, Module-Level failure management capability ensures all types of System Test functionalities whereas Unit-Level is capable to ensure Start-up Test and On-line Test.

- Rationale for 2.a & 2.b;

  Both of Set/Change Data Transmission Frequency and Data Transmission Interface features are modeled as optionally. However, Set/Change Data Frequency feature owes its presence to Data Transmission Interface as its functionality is offered under the content of this interface. Furthermore, in the scope of the MMLS Feature Model, there exists no other feature that owes its presence to Data Transmission Interface which leads to mutually-require relationship between these two features.

- Rationale for 3.a & 3.b;

  Set Resolution functionality can be achieved if only Monitor Keyboard Drawer feature is included in the product configuration since Monitor Keyboard Drawer is the only display unit alternative that can operate with any combination of two resolution alternatives. Additionally, Monitor Keyboard Drawer feature needs the presence of any alternative features included in Set Resolution variation point in order to perform its demonstrative functionalities properly. Therefore these two features mutually require each other in any product configuration.

- Rationale for 4.a, 4.b & 4c;

  Maintenance Interface offers user interface for the functionalities related with System Maintenance. If the optional features of System SW Update and Database Maintenance modeled under System Maintenance feature group are decided to be included in the configuration, they need the presence of Maintenance Interface capability of the software to perform their functionalities properly, considering their interaction with the user. For the System Test feature group, only Off-Line Test feature needs the presence of Maintenance Interface in order to be initialized by the user, whereas Start-up Test and On-line Test can initialize and operate without the user intervention. Therefore, requires relationship from Off-line Test feature, System SW Update and Database Maintenance feature groups to Maintenance Interface feature is needed to be defined.

- Rationale for 5.a, 5.b & 5.c;

  Wind Speed and Direction Sensor needs an initialization message from the MMLS software to have its activation parameter to be set to active, this initialization messaging facility is supported by Set Sensor Parameters feature. Moreover, in order to measure the direction of the wind correctly, reference measurement direction (in terms of degrees) with respect to the factory set measurement axis has to be set prior to its operation. This reference direction information with respect to factory set measurement axis can be set with the help of optional Set Sensor Parameters feature. Wind Speed and Direction Sensor is the only one that the system can offer the capability to set its internal parameters. Therefore inclusion of Set Sensor Parameters feature to the product configuration without the presence of Wind Speed and Direction Sensor feature has no meaning. Consequently, these two features mutually require each other.

  In extension to the reasoning above, Wind Speed and Direction Sensor is the only one that allows the system to ask for its operation status at any time. This sensor is capable to provide acknowledge messages as long as it is asked for status update through its interface. This capability is utilized by Off-line Test feature to let the user investigate the status of the sensor for proper operation. It should be noted that this capability can not be ensured by On-line Test, since On-line Test regards the periodic availability of messages from the sensors (simplex operation) to decide that they are operating properly, whereas Off-line Test is capable to send messages in order to receive availability messages from this sensor (duplex operation). As long as this sensor is included in the configuration, it needs the presence of Off-line Test capability.

- Rationale for 6.a, 6.b, 6.c & 6.d;
  Inkjet Printers that are modeled in MMLS domain are the former versions of Laser Printers. Therefore they are modeled to be in excludes dependencies with USB Port and Ethernet/LAN interface types which are of contemporary interface technologies and are not supported by Inkjet Printers. Similarly Laser Printers of MMLS domain does not support former classical interface technologies of Serial and Parallel Port types which lead to excludes dependencies from Laser Printers to these interface types.

- Rationale for 7.a, 7.b, 7.c & 7.d
  Wind Speed and Direction Sensor is compatible to only RS232 type of interface while Ambient Humidity and Temperature Sensor and Air Pressure Sensor are capable to transmit their signals in compliance with RS422 interface only. Since there exists no HW component having requirement for RS232 interface compatibility, Wind Speed and Direction Sensor and RS232 feature pair mutually require each other.

- Rationale for 8.a, 8.b & 8.c;
  LCD Display offered in the content of MMLS domain is compatible to only RS422 type of interface. Similarly, Monitor Keyboard Drawer unit needs

VGA interface to support its graphical interface applications. Due to the fact that there exists no feature other than Monitor Keyboard Drawer requiring VGA interface, VGA and Monitor Keyboard Drawer features are modeled to be in mutually-require relation with each other.

4.4. Integrating Complementary Quality Modeling with Functional Feature Model of MMLS Domain

In this section, as the beginning, the basics of complementary quality modeling is introduced. Subsequently, the basic constituents of the complementary model (namely the levels and operationalizations of feature oriented NFR analysis and the quantitative QAs of EFM) are developed. In following, these constituents are integrated to each other thereby ensuring the integration of the whole complementary quality model to the functional feature model of the MMLS domain**.**

4.4.1. Complementary Quality Modeling

On the way to perform feature oriented quality-based variability analysis of feature domains, the quality modeling knowledge inherent in the domains can be revealed and expressed by means of two different perspectives. As mentioned earlier, these perspectives are originated from two different aspects of quality modeling, namely the qualitative quality modeling aspects and the quantitative quality modeling aspects. Based on previously mentioned methodologies that the complementary quality modeling is inspired from; qualitative quality modeling is achieved by Feature-oriented NFR Analysis in which quality modeling knowledge is a kind of soft-goal to be achieved and possesses hard-to-quantify nature; quantitative quality modeling is achieved by building extensions on functional features to build up EFMs in which quality modeling knowledge is a fragment of a functional feature and can be expressed in terms of a quantifiable parameter.

Qualitative quality modeling aspects can be classified in different levels, each of which provides operationalizations that are some kind of functional feature projections of quality-based goals. This kind of quality modeling is beneficial in revealing level-based qualitative variability knowledge thereby enhancing the variability of the whole model. However, as it is integrated with a functional feature model (by means of direct relationships of either require or exclude), the operationalizations seem to lack the interaction with the functional features. This leads to a lack of control over functional features especially during AE phase, during which selections of features are realized to make up the product configurations. This is due to the reason that it is not always possible to construct direct relationships between all the operationalizations (or levels) provided by Feature-oriented NFR Analysis and the functional features existent in the domain model. Even if these relationships, in the form of implicit interdependencies, are constructed, in most cases, they run short of providing all possibilities of implicit interdependencies.

On the other hand, quantitative quality modeling aspects are the quality-based attributes belonging to specific functional features. This kind of quality modeling is beneficial in revealing the quantifiable quality-based attributes inherent in specific functional features. Nevertheless, from the variability point of view, these QAs seems to have no contribution to the variability of the whole model as long as they are not utilized during AE phase. The utilization of these QAs can be achieved by incorporating them within feature decision rules in the form of either require or exclude relationships with other features of the model.

The designation “complementary” is derived from the idea that these two different perspectives complement each other especially if the elicitation of variability knowledge inherent in feature domains is of concern. This complementary study finds its meaning by providing a two-way feedback between these two quality modeling perspectives in order to satisfy the shortcomings of both sides.

In its simplest explanation, following their elicitation, the quantitative QAs are connected to the levels (thereby to the operationalizations as well) derived by Feature-oriented NFR Analysis through the implicit interdependencies (i.e. requires and excludes relationships). In order to construct these relationships, each type of quantitative QA corresponds to a specific type of soft-goal derived by qualitative quality analysis. No implicit interdependencies other than these pairings (between the type of quantitative QAs and soft-goals) are allowed to be constructed.

In the scope of this work, as demonstrated in Figure 4.9 below, Complementary Quality Modeling approach together with its integration to MMLS functional feature model is implemented mainly in three phases. In the context of 1st phase, quantitative QA analysis is performed. In the 2nd phase, Feature-oriented NFR analysis is performed and types of soft-goals are specified in addition to the definition of operationalizations belonged to different levels of each soft-goal. In the last phase, quality modeling artifacts derived with the help of these two approaches are integrated to each other which also ensures the integration of the Complementary Quality Model to the Functional Feature Model. It should be noted that the Complementary Quality Modeling practice needs the existence of a functional feature model as a basis for its implementation prior to integration.

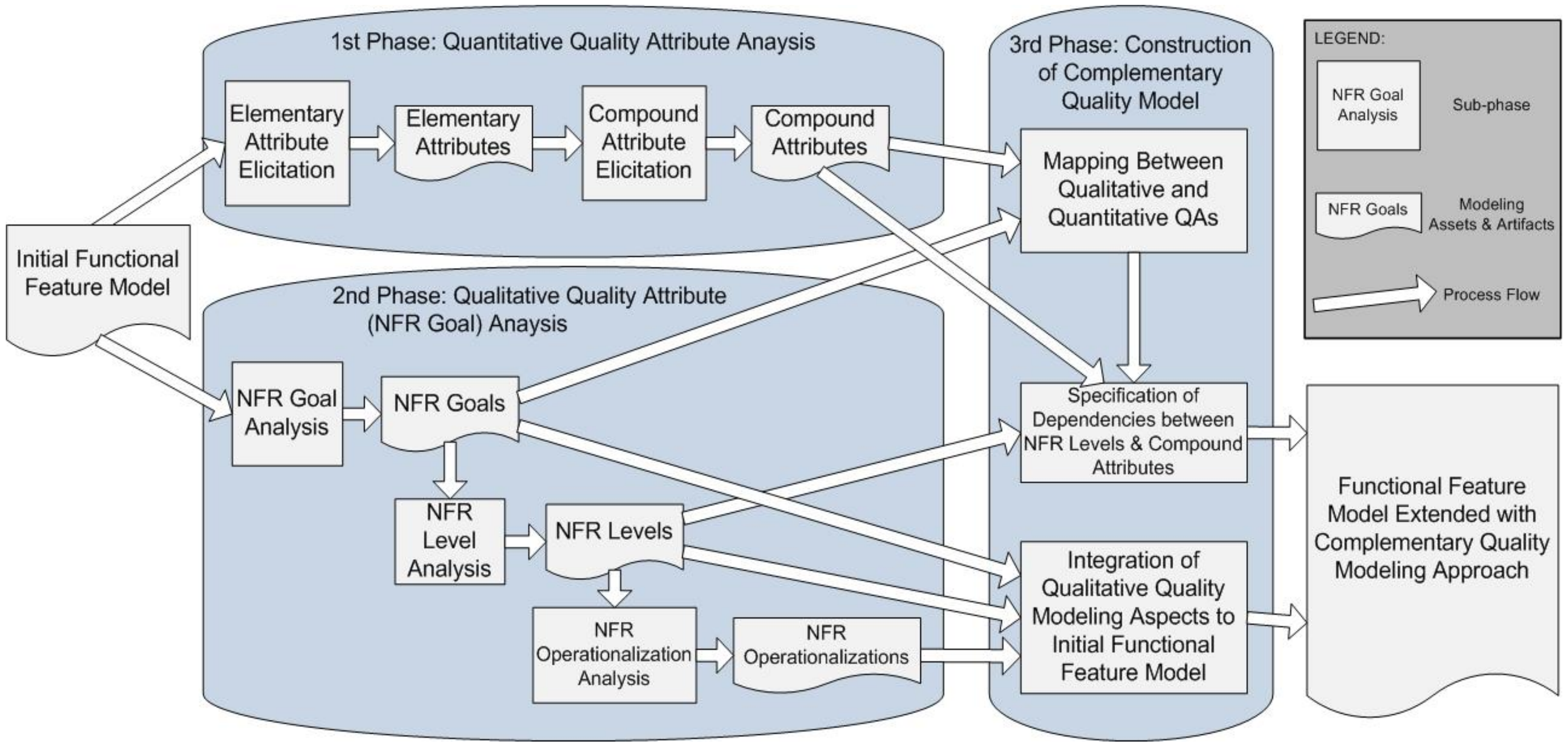

1st Phase: Quantitative Quality Attribute Anaysis
Elementary Attribute Elicitation
Elementary Attributes
Compound Attribute Elicitation
Compound Attributes
3rd Phase: Construction of Complementary Quality Model
Mapping Between Qualitative and Quantitative QAs
LEGEND:
NFR Goal Analysis
Sub-phase
NFR Goals
Modeling Assets & Artifacts
Process Flow
Initial Functional Feature Model
2nd Phase: Qualitative Quality Attribute (NFR Goal) Analysis
NFR Goal Analysis
NFR Goals
NFR Level Analysis
NFR Levels
NFR Operationalization Analysis
NFR Operationalizations
Specification of Dependencies between NFR Levels & Compound Attributes
Integration of Qualitative Quality Modeling Aspects to Initial Functional Feature Model
Functional Feature Model Extended with Complementary Quality Modeling Approach

Figure 4.9: Complementary Quality Modeling Process

4.4.2. 1st Phase: Quantitative Quality Attribute Analysis

In the scope of this phase, functional features involved in the initial functional feature model are examined so as to reveal their hidden quality-based attributes that has quantifiable nature.

These quality-based attributes are to be regarded as parameters that would have different values for different features. These parameters may belong to different domains (i.e. real, integer, boolean etc.) based on the set of values it can take. As the functional features are examined on the way to elicit their quantifiable QAs, care must be taken on the types of the attributes especially of the features included in the same feature groups. Possessing the same types of QAs is indispensable for the establishment of comparative relations among the options/alternatives offered in the same feature groups (to be clarified practically in the next section).

These quality-based attributes revealed from the child features of a specific feature group can be named as Elementary Attributes. An elementary attribute defined for a specific functional feature does not pose a significant importance on its own since it may or may not be included in product configuration. In order to derive more realistic reasonings from several elementary attributes (valid for different feature configurations), expressions has to be defined specifically for the parent nodes of these functional features, namely the feature groups (variation points). This expression should be in the form of a mathematical function whose variables are the attributes of its child features. This mathematical function may take any form depending on the usage of quality modeling context (i.e. total cost belonged to a parent feature is the sum of the costs belonged to its child features).

This expression can be named as Compound Attribute since it is composed of several elementary attributes or compound attributes owned by its child features. Similar to the case in elementary attributes, care must be taken on the type of compound attributes. That is to say, parent features have to possess compound attributes type of which is the same with elementary attributes belonged to its child features.

The values of compound attributes are dynamic such that they are dependent on the feature combination selected from the group of its child features. Different feature combinations of child features yield different compound attribute values for the parent features.

4.4.2.1. Elementary Attribute Elicitation

In the context of MMLS domain, as mentioned previously in *Section 4.3.2.1 Structural Interdependencies*, specification of hardware component configuration is said to put borderlines on the varied capabilities of the software components. This situation is handled by establishing implicit interdependencies between the SW and HW features.

However, these implicit interdependences fall short of responding the needs to manage the boundaries drawn by NFRs. As mentioned in *Section 3.1 Non-functional Requirements,* it is the non-functional concerns that define physical constraints on functional features. Therefore there should exist an extensive mechanism to elicit additional characteristics inherent in functional features that would have the expressive power in terms of quality concerns. As long as these characteristics are revealed and utilized systematically, non-functional (or quality) concerns can be managed efficiently.

At this place, the elementary attributes mined from functional features are the best candidates to be used for management of quality concerns. It must be noted that elementary attributes are utilized in a similar way with the establishment of implicit interdependencies realized in *Section 4.3.2.2 Implicit Interdependencies*.

In the context of MMLS domain, HW components feature group is the most suitable instance to reveal elementary attributes, as they can be regarded as sources of physical constraints due to their concrete nature. Besides they encapsulate abundant of characteristics possessing quantifiable characteristics (i.e. memory capacity of data storage units).

In the light of discussions provided above, four basic types of quantitative QAs (alias the elementary attributes) are defined in the context of Hardware Components feature group of MMLS domain, namely, capacity, cost, MTBF (Mean Time Between Failures) and DRT (Data Refreshment Time).

Capacity stands for the numerical value which expresses the memory storage capability of data storage units contained in MMLS domain. Capacity is meaningful for the Data Storage Units feature group. As demonstrated in *Figure 4.11*, in each variant of Data Storage Units feature group, capacity attribute is expressed in terms of GigaByte (GB) unit and is modeled in integer domain as the same type of parameter within each functional feature.

As all hardware components utilized in MMLS domain are COTS units, they all have a market price to be managed efficiently especially for the cases that overall production cost of the system is of concern. All the functional features taking part in HW components feature group contain cost as one of their elementary attribute and is expressed in terms of dollars ($). This attribute is modeled in integer domain as the same type of parameter throughout the features it is used.

MTBF is the numerical value representing the average time between the failures of hardware components. This attribute is a good sign of reliability and availability of the physical components thereby it is utilized by all features of HW components feature group. MTBF values for the features are expressed in terms of hours (hrs) and are modeled in integer domain as the same type of parameter within each functional feature. As will be clarified in *Section 4.4.2.2 Compound*

Attribute Elicitation, for the ease of computation in some exclusive cases when summation of more than one attribute is necessary, the features are decorated with the inverse of MTBF parameter (i.e. [$(\mathrm{MTBF})^{-1}$]).

DRT is the numerical value to express the elapsed time to transmit the refreshed data through the interfaces of HW components. It is expressed in terms of milisecond (ms) unit. This attribute is an indicator for the data transmission performance of HW components. DRT is modeled in the context of three HW components (namely the sensor units, display units and data storage units), those of which involve data transmission facilities. As referenced in *Figure 4.10* below, a single type of parameter (i.e. DRT) is defined to represent the transmission performance of each of these HW components. For the transmission between the sensor and the processor, data flow is simplex and has the direction from the sensor to processor. Therefore DRT parameter represented for this interface is dependent on the type of sensor. This is the reason why this parameter is defined as the elementary attribute of sensor units.

In a similar fashion, the same parameter is used to model the data transmission performance for the interface between the processor and the display units where the data flow is simplex and has the direction from processor to display units. Note that the value of DRT is dependent on the type of display. Therefore, this attribute is modeled as a part of the features in Displays feature group.

Different from the others, data flow between processor and the data storage units is duplex. Nevertheless DRT is the parameter to describe the data transmission performance of this interface and is dependent on the type of data storage unit utilized.

It must be noted with reference to *Figure 4.10* below that Sensors.DRT signifies overall DRT measure for sensor units defined in any of product configuration they are included in. The same is valid for the parameters of other HW components such as Displays.DRT and Data_Storage_Units.DRT.

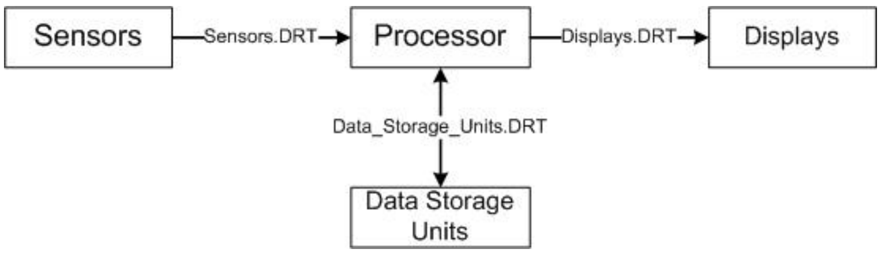
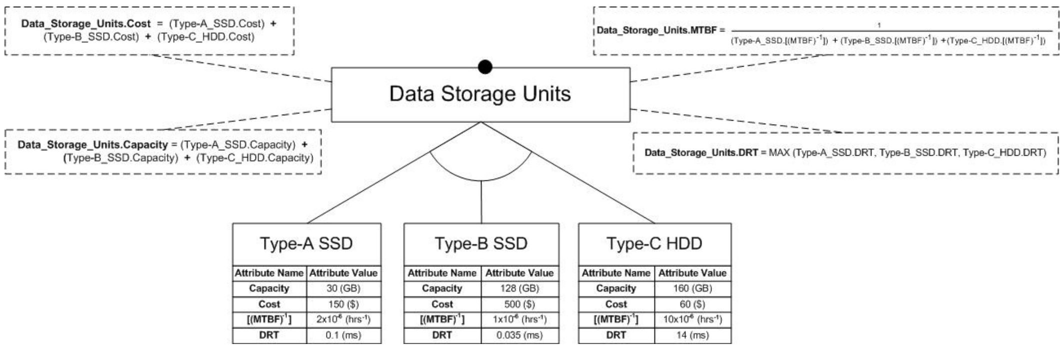


Figure 4.10: Hardware Interaction Based on Data Transmission Rates

With reference to Figure 4.10 above, it must also be noted that, first, the measured data provided from sensors is saved to database by means of Processor unit. Afterwards, this data is read from the database again to be finally fed to display units for presentation purposes under the control of Processor unit again. This is the typical route followed by every measured data sourced by sensor units.

Following the introductory discussion provided so far regarding the types and usage of the parameters to be utilized in the scope of the complementary quality modeling practice of MMLS domain, the rest of this section is dedicated to the details of the implementation. It must be noted that the numbers assigned to the elementary attributes of the features are common and tentative values based on the information collected from the market products. Besides, as presented in a number of instances through the rest of this section, it should be stated that the graphical notation adopted for the illustration of feature diagrams is inspired from the studies of Streitferdt et al. [42] and Benavides et al. [13].

As demonstrated in *Figure 4.11* below, all four types of elementary attributes are utilized in the context of Data Storage Units. This is a good instance to observe how a decision among different feature alternatives leads to several trade-offs among different aspects. It should be noted that the cost of the component is not directly proportional with the level of the capacity offered. It is reasonable to establish a proportionality relation between the cost and DRT or cost and MTBF parameters. Since HDDs include mechanical parts such as arms and spinning platters whereas SSDs includes none, HDDs are more susceptible to aging in the same operative conditions. This leads to a high level of MTBF values for SSDs compared to HDDs. As SSDs have the capability to realize parallel read and write operations on several partitions of the drive while HDDs can not, DRT assigned to SSDs are far lower than that of HDDs. Due to these reasons, cost is more likely to be related with the level of the technology involved in the component such that SSDs are state of the art types in data storage products. It can be concluded that performance comes with the higher cost. Furthermore as a different instance of trade-off regarding the outcome of a selection among the alternatives, HDDs offer higher level of capacity with higher level of cost whereas SDDs offer higher level of reliability and data transmission performance with higher cost.


Data_Storage_Units.Cost = (Type-A_SSD.Cost) + (Type-B_SSD.Cost) + (Type-C_HDD.Cost)
Data_Storage_Units.MTBF = 1 / (Type-A_SSD.[(MTBF)^-1]) + (Type-B_SSD.[(MTBF)^-1]) + (Type-C_HDD.[(MTBF)^-1])
Data Storage Units
Data_Storage_Units.Capacity = (Type-A_SSD.Capacity) + (Type-B_SSD.Capacity) + (Type-C_HDD.Capacity)
Data_Storage_Units.DRT = MAX (Type-A_SSD.DRT, Type-B_SSD.DRT, Type-C_HDD.DRT)
Type-A SSD
Type-B SSD
Type-C HDD

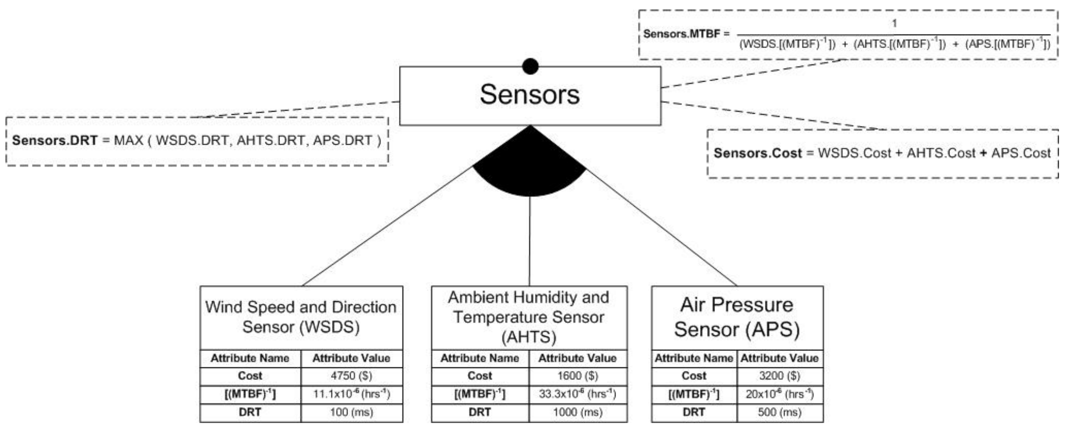
Type-A SSD	
Attribute Name	Attribute Value
Capacity	30 (GB)
Cost	150 ($)
$[(MTBF)^{-1}]$	$2x10^6$ (hrs^{-1})
DRT	0.1 (ms)

Type-B SSD	
Attribute Name	Attribute Value
Capacity	128 (GB)
Cost	500 ($)
$[(MTBF)^{-1}]$	$1x10^6$ (hrs^{-1})
DRT	0.035 (ms)

Type-C HDD	
Attribute Name	Attribute Value
Capacity	160 (GB)
Cost	60 ($)
$[(MTBF)^{-1}]$	$10x10^6$ (hrs^{-1})
DRT	14 (ms)

Figure 4.11: Extended Feature Diagram of Data Storage Units

As demonstrated in Figure 4.12 below, three types of elementary attributes are utilized to model the quantitative quality characteristics of the sensors, namely, cost, MTBF and DRT. It is not reasonable to derive proportionality relations these attributes since these characteristics are comparatively more typical (specific to type of products) with respect to other types of HW components.


Sensors.MTBF = 1 / ((WSDS.[(MTBF)^-1]) + (AHTS.[(MTBF)^-1]) + (APS.[(MTBF)^-1]))
Sensors
Sensors.DRT = MAX (WSDS.DRT, AHTS.DRT, APS.DRT)
Sensors.Cost = WSDS.Cost + AHTS.Cost + APS.Cost
Wind Speed and Direction Sensor (WSDS)
Ambient Humidity and Temperature Sensor (AHTS)
Air Pressure Sensor (APS)

Wind Speed and Direction Sensor (WSDS)

Attribute Name	Attribute Value
Cost	4750 ($)
$[(MTBF)^{-1}]$	11.1×10^{-6} (hrs^{-1})
DRT	100 (ms)

Ambient Humidity and Temperature Sensor (AHTS)

Attribute Name	Attribute Value
Cost	1600 ($)
$[(MTBF)^{-1}]$	33.3×10^{-6} (hrs^{-1})
DRT	1000 (ms)

Air Pressure Sensor (APS)

Attribute Name	Attribute Value
Cost	3200 ($)
$[(MTBF)^{-1}]$	20×10^{-6} (hrs^{-1})
DRT	500 (ms)

Figure 4.12: Extended Feature Diagram of Sensors

As referenced in Figure 4.13 below, similar to the case for sensors, three types of elementary attributes, namely cost, MTBF and DRT are utilized. Typical values assigned to cost parameter indicate that Monitor Keyboard Drawer possesses almost twice as much price as the LCD display. This is a usual situation concerning the user-friendly facilities (i.e. interaction with the user) offered by Monitor Keyboard Drawer which is not valid for LCD Display. As is the case in Data Storage Units, once again, higher level of services for user satisfaction comes along with higher costs. Again there exists a trade-off between the cost and MTBF as is the case for Data Storage Unit alternatives. But this time there exists an inverse proportionality between them such that LCD Display with higher MTBF value is offered with lower cost. The reason why LCD owns twice as much MTBF value as Monitor Keyboard Drawer is possibly due to the comparatively more number of components are involved in Monitor Keyboard Drawer. It should be noted that, based on the basic formula of overall MTBF calculation, increasing the number of components with the same MTBF values even reduces the total MTBF of all comprised components. However higher cost compensates for the need of higher data transmission performance by means of its sophisticated configuration.

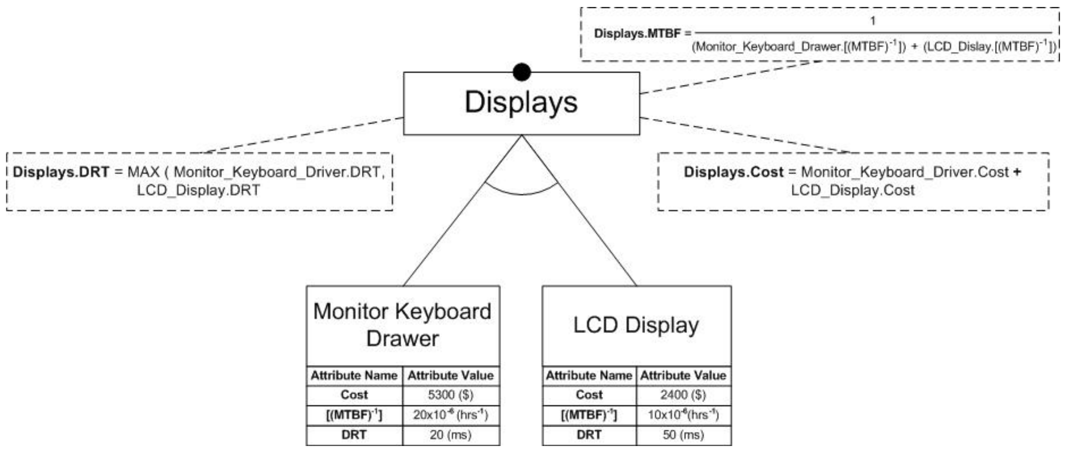

Displays.MTBF = 1 / (Monitor_Keyboard_Drawer.[(MTBF)^-1]) + (LCD_Dislay.[(MTBF)^-1])
Displays
Displays.DRT = MAX (Monitor_Keyboard_Driver.DRT, LCD_Display.DRT
Displays.Cost = Monitor_Keyboard_Driver.Cost + LCD_Display.Cost
Monitor Keyboard Drawer
Attribute Name | Attribute Value
Cost | 5300 ($)
[(MTBF)^-1] | 20x10^-6 (hrs^-1)
DRT | 20 (ms)
LCD Display
Attribute Name | Attribute Value
Cost | 2400 ($)
[(MTBF)^-1] | 10x10^-6 (hrs^-1)
DRT | 50 (ms)

Figure 4.13: Extended Feature Diagram of Displays

In reference to Figure 4.14, Laser Printers offer slightly higher levels of reliability with respect to Inkjet Printers which requires comparatively higher values for cost.

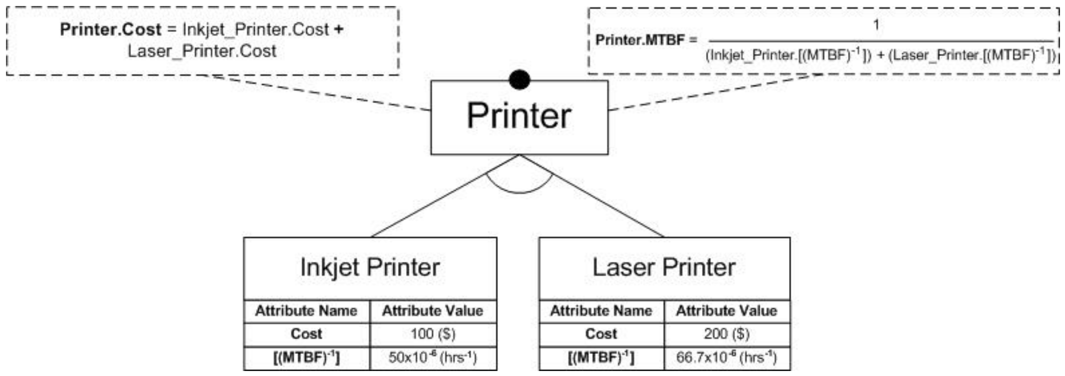

Printer.Cost = Inkjet_Printer.Cost + Laser_Printer.Cost
Printer.MTBF = 1 / (Inkjet_Printer.[(MTBF)^-1]) + (Laser_Printer.[(MTBF)^-1])
Printer
Inkjet Printer
Attribute Name | Attribute Value
Cost | 100 ($)
[(MTBF)^-1] | 50x10^-6 (hrs^-1)
Laser Printer
Attribute Name | Attribute Value
Cost | 200 ($)
[(MTBF)^-1] | 66.7x10^-6 (hrs^-1)

Figure 4.14: Extended Feature Diagram of Printer

4.4.2.2. Compound Attribute Elicitation

In the previous section, decoration of leaf features with elementary attributes has been realized. This section is dedicated to explain how the owner of these leaf features, namely the parent features (i.e. feature groups) are decorated with quantitative QAs which will be named as compound attributes from now on.

Compound attributes owned by the feature groups are different from elementary attributes as they can not be represented by a single parameter on their own. They are dependent on the feature configuration that would be selected for any product instance. So as to represent any selectable feature configuration, they should be in the form of flexible aggregative expressions. In order to cope with the numerical values represented by the parameters, these aggregative expressions should be in the form of suitable mathematical functions.

These mathematical functions are based on the type of the parameters they utilize. As mentioned before, each type of the parameter is based on a specific quality concern (i.e. MTBF parameter is based on reliability concern). Therefore these mathematical functions possess the same type of quality concern with their parameters.

Before specifying the details regarding the aggregative functions for compound attributes, an exceptional case regarding the utilization of elementary attribute values into compound attributes has to be clarified. The exceptional case is caused by the absence of an elementary attribute in the compound attribute expression. This is due to the fact that the owner of the elementary attribute (namely the leaf feature) is not included in the selected product configuration. This brings the need of definition for null value of zero for the elementary attribute of a feature which is not included in the configuration. The representative of these features (namely the elementary attributes) in the compound attribute expression takes the null value of zero.

The aggregation function adopted to express compound attribute for cost is based on the summation of its parameters, namely the elementary attributes.

$$Cost_{aggregated} = Cost_1 + Cost_2 + Cost_3 + ... \quad (9)$$

Similarly, aggregation function owned by compound attribute of capacity is based on summation operation.

$$Capacity_{aggregated} = Capacity_1 + Capacity_2 + Capacity_3 + \dots \quad (10)$$

Calculation of compound attribute for MTBF is based on its well-known typical formula as given in (11) below;

$$MTBF_{aggregated} = \frac{1}{\frac{1}{MTBF_1} + \frac{1}{MTBF_2} + \frac{1}{MTBF_3} + \dots} \quad (11)$$

It must be noted that, as can be observed in *Figure 4.11*, *Figure 4.12*, *Figure 4.13* and *Figure 4.14*, the elementary attribute of [(MTBF)$^{-1}$] is utilized instead of MTBF value itself. This is a precaution for the cases that a feature is not included in the configuration and its representative contributes to the compound attribute expression with null value. If it was directly imported in the expression of (11), the total MTBF would come out to be 0, as the denominator of the equation diverges to infinity. This yields to completely wrong and useless results. However, in such cases, [(MTBF)$^{-1}$] takes the null value and the total MTBF calculation is not effected from this exceptional case.

The assignment of null values to elementary attributes due to non-existence of their owner feature is valid for all types of compound attribute calculations.

Different from aforementioned compound attribute expressions, DRT for compound attributes is based on a different calculation procedure. The rationale behind the calculation of this attribute expression is based on the minimum rule such that the pace of data transmission between component groups is perceived to be as fast as the slowest one, as inspired from the famous phrase “the chain is as strong as the weakest ring”. For instance, the performance of the sensors in a sample product is defined with respect to the one with the minimum performance.

Similarly, considering the display units, the speed of data transmission on the line between the processor and display unit can be specified by filtering out the one with the lowest speed. For the case of DRT attributes, higher values of DRT imply higher values of delay are needed to feed the line with the refreshed data. Therefore, we can conclude that higher DRT means lower data transmission performance. This is the reason why MAX operand is utilized on the way to specify the DRT with lowest data transmission performance, as referenced in expression (12) below.

$$DRT_{aggregated} = MAX\ (DRT_1, DRT_2\ , DRT_3 ...) \quad (12)$$

Same type of parameters belonged to different features (as demonstrated in *Figure 4.11*, *Figure 4.12*, *Figure 4.13* and *Figure 4.14*) are aggregated with appropriate functions given in (9), (10), (11) and (12) to build up compound attributes.

Considering the whole system, as referenced in *Figure 4.15* below, expressions to reflect the overall quantitative quality characteristics of the system has to be provided. For the sake of convenience of notation, these attributes are designated with overall phrases.

The compound attributes of Overall Capacity, Overall Cost and Overall MTBF are expressed with the same formulas given in (9), (10) and (11). From this point of view, they can be regarded as function of functions. DRT is aggregated based on a formula different from the one given in (12). With reference to *Figure 4.10*, considering the whole system, the trip of a typical measured data from source (sensors) to target (display units) is as follows: transmission between sensor and processor is directional from sensors to processor; it is followed by bi-directional (i.e. first the data is written to database, then it is read) transmission between processor and data storage units; finally it ends with one-way transmission between processor and display units. All these transmissions are realized in a consecutive order such that, similar like estimating total latency through transmission lines, overall DRT value can be aggregated by summing DRT values assigned to each transmission line, as referenced in expression (13) below. It should be noted that, DRT attribute belonged to data storage unit contributes to expression with twice of its value due to bi-directional characteristic of the transmission line.

$$Overall\ DRT = Sensors.DRT + Displays.DRT + 2\ x\ (Data\_Storage\_Units.DRT) \quad (13)$$

Besides, this is the final step of 1st phase since overall aggregated values for specific quality concerns are extracted from their owner features. As explained in the next sections of this chapter, these overall values will be utilized to provide reasoning about the quality concerns.

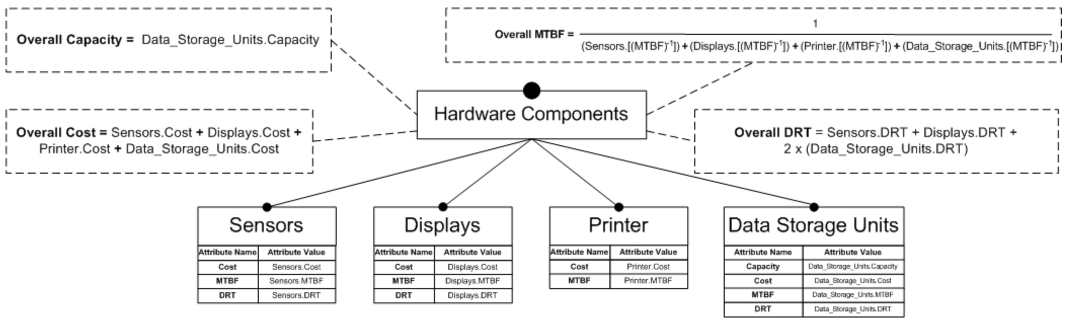

Overall Capacity = Data_Storage_Units.Capacity
Overall MTBF = 1 / (Sensors.[(MTBF)^-1]) + (Displays.[(MTBF)^-1]) + (Printer.[(MTBF)^-1]) + (Data_Storage_Units.[(MTBF)^-1])
Overall Cost = Sensors.Cost + Displays.Cost + Printer.Cost + Data_Storage_Units.Cost
Hardware Components
Overall DRT = Sensors.DRT + Displays.DRT + 2 x (Data_Storage_Units.DRT)
Sensors
Attribute Name | Attribute Value
Cost | Sensors.Cost
MTBF | Sensors.MTBF
DRT | Sensors.DRT
Displays
Attribute Name | Attribute Value
Cost | Displays.Cost
MTBF | Displays.MTBF
DRT | Displays.DRT
Printer
Attribute Name | Attribute Value
Cost | Printer.Cost
MTBF | Printer.MTBF
Data Storage Units
Attribute Name | Attribute Value
Capacity | Data_Storage_Units.Capacity
Cost | Data_Storage_Units.Cost
MTBF | Data_Storage_Units.MTBF
DRT | Data_Storage_Units.DRT

Figure 4.15: Extended Feature Diagram of Hardware Components

With reference to Figure 4.15 above, it should be noted that the parameters given in the “Attribute Value” column of each feature group is dependent on the selection of the feature combination from each respective feature groups. For instance, Sensors.Cost value has different values for different feature configurations specified from Sensors feature group. Besides, different from the case in elementary attributes, the parameters in “Attribute Value” column can never take null values since the feature group they are belonged to possesses at least one feature included in any sample of product configuration. This is the reason why MTBF parameter is assigned to feature groups as itself instead of its inverse.

4.4.3. 2nd Phase: Qualitative Quality Attribute (NFR Goal) Analysis

In the scope of this phase, departing from the knowledge and inspiration sourced by the types of quantitative QAs; aims, concerns and purposes of the stakeholders (i.e. system developers or users) are analyzed. On the way to achieve this analysis, NFR goal model is adopted.
The structure adopted for the construction of the NFR goal model is mainly inspired from the study of Peng et al. [10] in addition to its origins as the previous studies performed by Chung et al. [19] and Cysneiros and Leite [43].

The basic structure of the NFR model is as follows. Each NFR goal is decomposed into three levels as low, medium and high. Introducing the decomposition of NFR goals into several levels is favorable in terms of modeling qualitative quality concerns where quantification on quality concerns is not possible, advantageous or meaningful.

Each of these levels are decorated with appropriate operationalizations to be treated as selectable features since they are some kind of functional attribute projections of non-functional concerns. The levels and thereby the operationalizations are modeled as alternative to each other meaning that only one level is allowed to be selected for each sample of product configuration. This provides several benefits as follows. Modeling with at least three alternatives improves overall variability performance of the whole model. Moreover, thanks to this alternative relation among the various levels, different combinations of trade-off selections among NFR goals become available (i.e. an instance of product with Low Resource Utilization, Medium Accuracy, High Availability and Low Cost of Ownership).

The types of quantitative quality aspects revealed in the previous section specify the starting point to seek the probable aims that the system should achieve. In the scope of the MMLS domain, this brings the need for definition of four types of NFR goals, namely Resource Allocation, Accuracy, Availability and Cost of Ownership. As referenced in *Figure 4.16* below, these NFR goals can be described as root goals since they are modeled as mandatory feature groups at the uppermost level of the initial feature model. The reason why these root goals are modeled as mandatory can be specified as follows. NFR goals are different from the functional features due to their vague nature such that a functional feature can be clearly defined to be involved or not in a product configuration. But the existence or non-existence of goals originated from non-functional (or quality) concerns can not be defined exactly such that any type of quality concern may take part in any configuration to some extend. However, their contribution in terms of various levels can be utilized in product configurations. Below a detailed discussion on the origins of these NFR goals are provided together with their operationalizations assigned to each level.

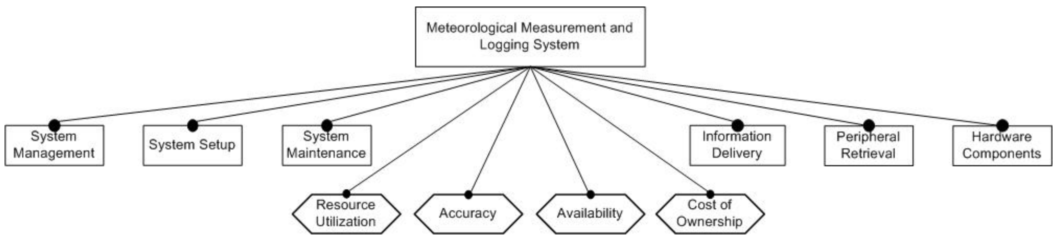


Figure 4.16: NFR Goal Integrated Feature Diagram of MMLS Domain – Uppermost Level

Resource Utilization is one of the vital indicators of Efficiency of a software product. It is defined as the capability to provide appropriate performance, relative to the amount of resources used, under specified conditions. Resource Utilization is defined as the capability of the same software product to manage and utilize these relative amounts of resources as the software performs its function. [44] In the light of these quality model definitions stated in

ISO/IEC9126 standard, the levels of Resource Utilization NFR goal is prescribed as depicted in *Figure 4.17* below. With reference to *Figure 4.17*, as the level of the goal is escalated, the capability offered by the system in terms of storing and managing higher amounts of data is improved as well. For instance, inclusion of low level Resource Utilization allows the software to record only the raw data without attachments of time and source information. However, a high level of Resource Utilization means that data stored to data storage units includes the recording date/time in addition to the source information of the raw data. Naturally, this high level of resource management comes with its toll such that higher amounts of data storage units are needed.

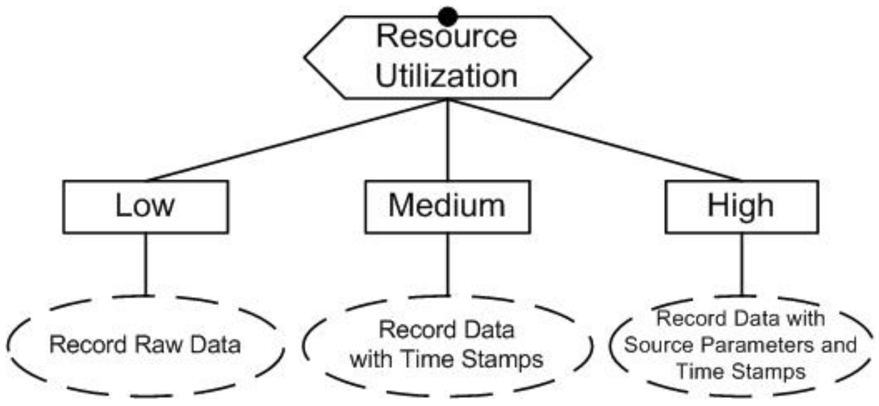

Resource Utilization
Low
Medium
High
Record Raw Data
Record Data with Time Stamps
Record Data with Source Parameters and Time Stamps

Figure 4.17: NFR Graph for Resource Utilization

As demonstrated in Table 4.2 below, low and medium level operationalizations of Resource Utilization need further implicit interdependencies. Log Operational Information feature modeled under System Log Management feature group excludes the low and medium level operationalizations of Resource Utilization NFR goal. This is due to the fact that Log Operational Information feature requires both the time and source information to perform its basic missions of keeping track of log-in – log-out information, changes in the status of system SW and measured data exchanges between sensors and external systems.

Table 4.2: Implicit Interdependencies Between Resource Utilization NFR Goal and System Log Management Feature Group

Source Feature	Dependency Relationship	Target Feature
Log Operational Information	Excludes	Record Raw Data
	Excludes	Record Data with Time Stamps

Being one of the significant factors of Functionality concern, Accuracy is defined as the capability of a software product to present correct results with required precision. [44] Precision for the perceived results is directly proportional with the refreshment frequency of the data provided to indicator units through the appropriate interfaces. In the scope of the MMLS domain, the user interface for perception of accuracy and precision is sustained by means of display units. In addition to display units, different type of the sensors and data storage units utilized in the product configuration are the other actors effecting the data refreshment frequency thereby the level of accuracy as well. In the domain of display units, the accuracy of the presented data is specified as the sensitivity. Therefore, with the assumption of the same set of correct data transmitted with different refresh rates causes different perception levels of sensitivity; as NFR goal of Accuracy, three levels of sensitivity with percentages of 10, 5 and 1 are offered for the low, medium and high levels of accuracy respectively.

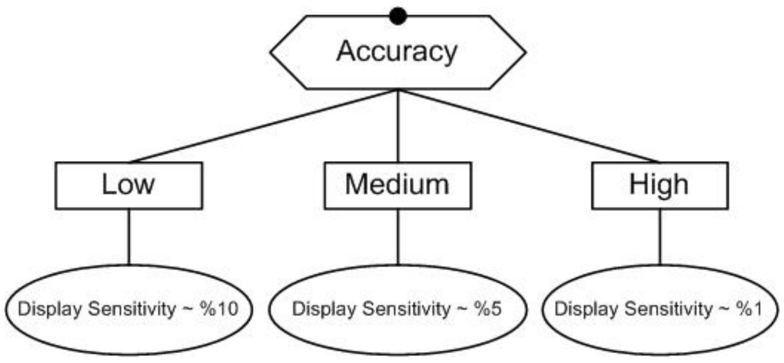


Figure 4.18: NFR Graph for Accuracy

As an important parameter of Reliability concern, Availability is the capability of a software product which ensures the satisfactory performance of a required function during a specified duration of time. [44] On the way to specify a system in terms of its availability against probable failures, the failure rates of its constituents are the determinants for failure avoidance capability. Being encouraged from this variable capability depending on different instances of MMLS configurations, three levels of availability operationalizations are offered. These levels are classified on the basis of different durations needed for periodical maintenance of the system. For example, an instance of MMLS with low overall MTBF value is said to possess low level of failure avoidance leading to low level of availability. This brings the need of periodic maintenance in a more frequent fashion. Selection of low level Availability offers (and requires at the same time) four times of periodic maintenance per year whereas high level offers/requires once a year.

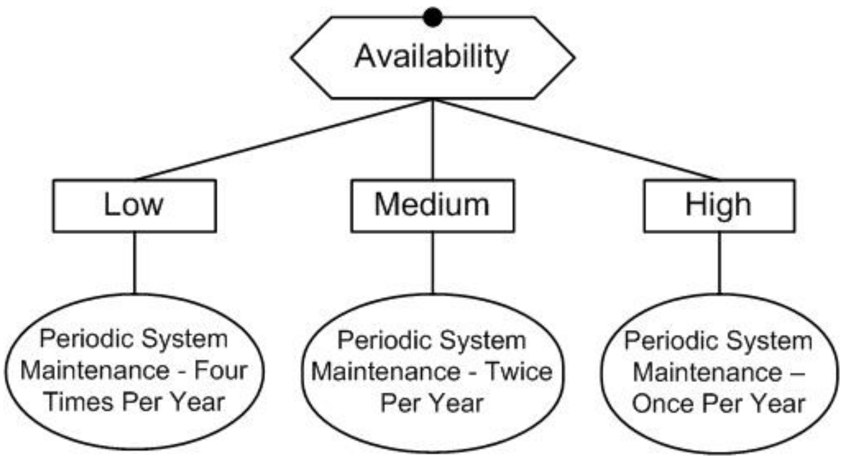

Availability
Low
Medium
High
Periodic System Maintenance - Four Times Per Year
Periodic System Maintenance - Twice Per Year
Periodic System Maintenance – Once Per Year

Figure 4.19: NFR Graph for Availability

Different from other NFR goals mentioned so far, Cost of Ownership is based on a different rationale. Cost of Ownership quality is inspired from a financial estimate named as Total Cost of Ownership (TCO). For computer systems, it is defined as “the total of direct capital investment in hardware and software including indirect costs of installation, training, repairs, downtime, technical support, and upgrading.” [45] It must be noted that resources provided especially for training, repairment and technical support required by the system are vital in terms of long-term system availability throughout its life-cycle. In the scope of the MMLS domain, direct capital investment in hardware is modeled in the context of quantitative quality modeling practice provided in *Section 4.4.2* as the 1st phase of Complementary Quality Modeling approach. Being the operationalizations assigned to different levels of NFR goal named Cost of Ownership, different combinations of indirect costs for training, technical support and repairs (in the form of warranty and spare-parts) are offered to the user of the system. Roughly speaking, components with higher MTBF values possess higher levels of costs meaning that higher costs during the production of the system bring better availability in the long-term. Therefore, as referenced in Figure 4.20 below, if the cost of ownership is selected to be high for any instance of product, technical supports of warranty and spare parts are not offered. But a system with low Cost of Ownership is more vulnerable to failures. From the user’s point of view, this leads to the need of technical support in the form of 2-year warranty in addition to the delivery of spare parts that are replaceable by end-user in cases of failures.

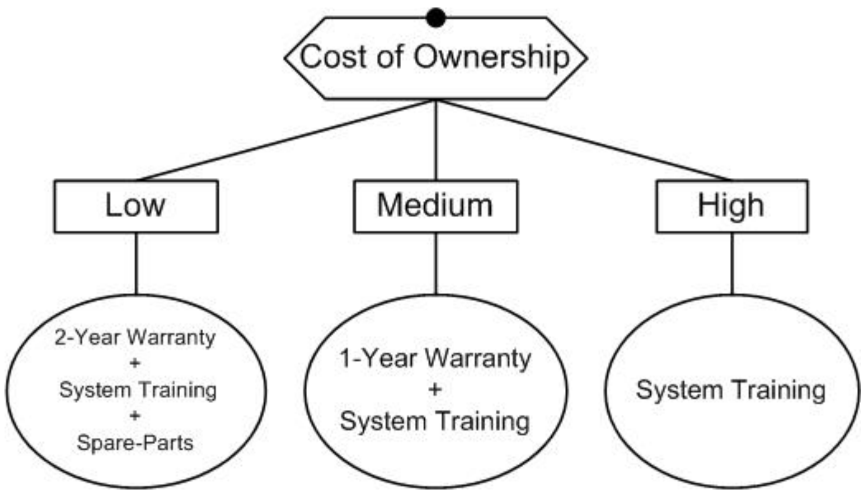

Cost of Ownership
Low
Medium
High
2-Year Warranty + System Training + Spare-Parts
1-Year Warranty + System Training
System Training

Figure 4.20: NFR Graph for Cost of Ownership

It should be noted that, notation as inspired from [10] for designation of operationalizations is as follows: Dashed circles as provided in Figure 4.17 represent dynamic operationalizations, those of which requires an action to be realized. However, solid circles with instances in Figure 4.18, Figure 4.19 and Figure 4.20 stand for static operationalizations meaning a restriction on the system capability if utilized in system configuration.

4.4.4. 3rd Phase: Construction of the Complementary Quality Model

In this phase, in its basic specification, the connections between quantitative and qualitative attributes, elicited in 1st and 2nd phases respectively, are established. As demonstrated in Table 4.3, each type of quantitative QA has its counterpart in the form of qualitative QA.

Table 4.3: Mapping Between Quantitative and Qualitative QAs

Type of Quantitative Quality Attribute (Compound Attributes)	Type of Qualitative Quality Attribute (NFR goals)	Affected Quality Modeling Concern
Overall Cost	Cost of Ownership	Efficiency
Overall Capacity	Resource Utilization	
Overall MTBF	Availability	Reliability
Overall DRT	Accuracy	Functionality

The connections between qualitative and quantitative QAs are to be realized in the form of excludes dependencies. These dependencies are bi-directional relationships defined between the operationalizations (belonged to different levels of NFR goals) and Hardware Components feature group. The rules and conditions for these exclude relationships are defined with respect to compound attribute values assigned to Hardware Components root node.

As demonstrated in *Table 4.4*, excludes relationships between Hardware Components and Cost of Ownership levels are realized depending on the values owned by Overall Cost compound attribute. The same dependency rationale is adopted for Resource Utilization levels through Overall Capacity attribute values, for Availability levels through Overall MTBF values and finally for Accuracy levels through Overall DRT values.

The meaning and purpose of all these relations become apparent when it is time to select any one of the available levels of the NFR goals as part of product configuration. As mentioned below, the benefit of selection out of these levels is two fold:

- Each of the levels assigned to four different types of NFR-goals serves different operationalizations available to selection for their contribution to product configuration. This is beneficial in the sense that overall variability of the feature model is enhanced by these functional domain projections of basic quality concerns. In other words, non-functional variability of the whole model is enhanced.
- More importantly, selections among various levels of NFR-goals (possibly to be realized during AE phase) has a filtering effect over the feature alternatives available under Hardware Components feature group. This filtering operation, which is realized by excludes relationships, limits the available feature alternatives to be selected as the variants and sub-variants of Hardware Components variation point. By means of these relationships defined, the maturity of the feature model is enhanced as it provides more realistic feature combinations for available product configurations. Further discussion on this enhancement is provided in the scope of Chapter 5.

The rationale behind the rules for excludes relationships are based on classification of numerical ranges owned by each of the compound attributes. Each level of NFR-goals are mapped to a compliant range of values specified for their counterpart compound attribute. For instance, as demonstrated in Table 4.4 below, Low level Cost of Ownership is compliant to the range of Overall Cost values between 4100 and 7825 $. This means, if Low Cost of Ownership is included in the product configuration, by means of excludes relationships, only the product configurations compliant with that specified range of overall cost values are available for selection.

Table 4.4: Traceability Between Range of Quantitative QA Values and Levels of Qualitative QAs

Quantitative QAs (Compound Attributes)		**Qualitative QAs (NFR goals)**	
Type of Quantitative QA	**Range of Values for Quantitative QAs**	**Level of Qualitative QA**	**Type of Qualitative QA**
Overall Cost ($)	$4100 < Cost_{overall} < 7825$	Low	**Cost of Ownership**
	$7825 < Cost_{overall} < 10725$	Medium	
	$10725 < Cost_{overall} < 15600$	High	
Overall Capacity (GB)	$Capacity_{overall} = 30$	Low	**Resource Utilization**
	$Capacity_{overall} = 128$	Medium	
	$Capacity_{overall} = 160$	High	
Overall MTBF (Hours)	$6200 < MTBF_{overall} < 8100$	Low	**Availability**
	$8100 < MTBF_{overall} < 9500$	Medium	
	$9500 < MTBF_{overall} < 14000$	High	
Overall DRT (ms)	$1050 < DRT_{overall} < 1080$	Low	**Accuracy**
	$550{,}1 < DRT_{overall} <1050$	Medium	
	$120 < DRT_{overall} <550{,}1$	High	

For further clarification, all the exclude relationships defined between the levels and Hardware Components feature group are provided in *Table 4.5* below. For instance, High Cost of Ownership excludes a feature configuration if the condition ‘$Cost_{overall} < 10725$’ is satisfied. Similarly, Medium Cost of Ownership excludes any feature configuration that satisfies the conditions of either ‘$4100 < Cost_{overall} < 7825$’ or ‘$10725 < Cost_{overall} < 15600$’. Obviously, the limits suggested here are all tentative, and may be adjusted by domain experts as necessary. The aim here has been to separate Qualitative QA levels uniformly into regions with respect to the number of available configurations derived from Hardware Components feature group. This separation rationale is utilized in the excludes relationships as shown below in Table 4.5.

Table 4.5: Specification of Excludes Relationships

NFR Goal	Levels	Rules for Exclude Relationships
Cost of Ownership	Low	$7825 < \text{Cost}_{\text{overall}}$
	Medium	$4100 < \text{Cost}_{\text{overall}} < 7825$ **OR** $10725 < \text{Cost}_{\text{overall}} < 15600$
	High	$\text{Cost}_{\text{overall}} < 10725$
Resource Utilization	Low	$\text{Capacity}_{\text{overall}} = 128$ **OR** $\text{Capacity}_{\text{overall}} = 160$
	Medium	$\text{Capacity}_{\text{overall}} = 30$ **OR** $\text{Capacity}_{\text{overall}} = 160$
	High	$\text{Capacity}_{\text{overall}} = 30$ **OR** $\text{Capacity}_{\text{overall}} = 128$
Availability	Low	$8100 < \text{MTBF}_{\text{overall}}$
	Medium	$6200 < \text{MTBF}_{\text{overall}} < 8100$ **OR** $9500 < \text{MTBF}_{\text{overall}} < 14000$
	High	$\text{MTBF}_{\text{overall}} < 9500$
Accuracy	Low	$\text{DRT}_{\text{overall}} < 1050$
	Medium	$120 < \text{DRT}_{\text{overall}} < 550{,}1$ **OR** $1050 < \text{DRT}_{\text{overall}} < 1080$
	High	$550{,}1 < \text{DRT}_{\text{overall}}$

Table 4.6 below demonstrates some available feature configurations to be utilized in product configurations associated with some instances of Qualitative QA level configurations.

Table 4.6: Effect of Qualitative QAs on Functional Feature Configurations

Qualitative QA Configurations				**Functional Feature Configurations Filtered Out from Hardware Components Feature Group**
Cost of Ownership	**Resource Utilization**	**Availability**	**Accuracy**	
LOW	HIGH	MEDIUM	MEDIUM	{AHTS, Type-C HDD, Monitor Keyboard Drawer, Inkjet Printer}, {APS, Type-C HDD, LCD Display, Laser Printer}
LOW	MEDIUM	HIGH	LOW	{AHTS, Type-B SSD, LCD Display, Inkjet Printer}
MEDIUM	HIGH	LOW	MEDIUM	{AHTS, APS, Type-C HDD, Monitor Keyboard Drawer, Inkjet Printer}, { AHTS, APS, Type-C HDD, Monitor Keyboard Drawer, Laser Printer}, {WSDS, AHTS, Type-C HDD, LCD Display, Laser Printer}
HIGH	LOW	MEDIUM	HIGH	{WSDS, APS, Type-A SSD, Monitor Keyboard Drawer, Laser Printer}
LOW	LOW	LOW	LOW	{AHTS, APS, Type-A SSD, LCD Display, Laser Printer}
MEDIUM	MEDIUM	MEDIUM	MEDIUM	NO AVAILABLE PRODUCT CONFIGURATION
HIGH	HIGH	HIGH	HIGH	NO AVAILABLE PRODUCT CONFIGURATION

It must be noted that for some combinations of Qualitative QAs, the Complementary Quality Attribute Model comes up with no available product configuration. This is due to the trade-off relationship between different types of

Qualitative QAs based on the selections among their respective levels. For instance, as referenced in Table 4.6 above, selection of High level for all Qualitative QAs leads to no available product configuration. This is an evidence for the idea that High level of Cost of Ownership does not guarantee the high levels for other Qualitative QAs. Possible causes of this situation can be listed as follows:

Type-B SSD can offer Medium level of Resource Utilization with High level of Cost of Ownership such that high level of overall cost compensates for high level of Availability, not Resource Utilization. Besides, High level of Cost of Ownership ensures the highest number of features available for selection out of Hardware Components feature group (i.e. all types of sensors can be included in a product configuration if and only if High Cost of Ownership is included in the product configuration). But this has a diminishing effect on overall MTBF value leading to Low Availability (i.e. configurations of High Cost of Ownership with all sensor types are all limited to Low Availability selection).

CHAPTER 5

EVALUATION OF THE COMPLEMENTARY QUALITY MODELING APPROACH

5.1. Introduction

In this chapter, most of the criteria introduced in Chapter-2 find their meaning as the evaluations are realized with respect to the relative feature model assets that are subject of comparison. Results of these criteria are first obtained for the initial functional feature model, it is followed by collection of same type of results from the same model extended with qualitative QAs. Finally, acquisition of evaluation data is completed by gathering the results of the same model extended with Complementary Quality Modeling approach.

In the course of presenting comparative discussions on several evaluation criteria, for the sake of convenience of expressions for the feature models:

- Initial Functional Feature Model is denoted as Model-1;
- Functional Feature Model extended with Qualitative QAs (NFR Goals) is denoted as Model-2;
- Functional Feature Model extended with Complementary Quality Modeling Approach is denoted as Model-3.

5.2. Evaluation of Complementary Quality Modeling Approach

On the way to provide a comparative assessment and analysis related with the benefits and drawbacks of the proposed work, the results for the criteria of

FIPAP, VF versus ECR, RPER, commonality and DoO are presented in following sub-sections with supportive ideas relevant to their contribution to Non-functional Variability Management. The criterion of homogeneity is not subject to evaluation due to the non-existence of a unique feature in any combination of available products.

In the scope of this chapter, the aim is to assess the feature models with respect to quantifiable and comparable criteria. A quality evaluation of the developed software products is not intended. For this reason, during the selection and collection of evaluation criteria, the metrics are selected in a way that their parameters (i.e. the number of features, number of products, number of relationships between features etc.) can easily be collected from the respective feature models of comparison. By this way, the subjective effects of criteria are aimed to be suppressed.

As mentioned in Chapter-4, during definition and elicitation of NFR goals, ISO/IEC9126 standard is used as reference and inspiration which consists of many software quality assessment metrics as well. But these metrics deal directly with the software product itself, not the model of concern. That is why the metrics utilized in the scope of comparative evaluation of feature models are identified exclusive of this standard.

Besides, the ISO/IEC9126 standard simply aims to reason and estimate the impacts of quality concerns on a software product whose development process is completed, with components ready for testing. However what is aimed in this study is to evaluate the contribution of Complementary Quality Modeling approach in the scope of Domain Analysis phase with respect to general quantifiable SPL-oriented metrics.

5.2.1. Evaluation in terms of FIPAP Criterion

As referenced in *Section 2.3.4.7*, FIPAP, originally defined by Kasikci [38], is a significant indicator for the feature interaction problem avoidance capability of a model. FIPAP basically measures the ability of a model in exploring and exposing the possibility of encountering feature interaction problems which originate from lack of necessary exclude relationships.

As mentioned earlier, an increase in the number of features involved in a model leads to an increase in the number of feature interaction problems. Concerning the implementation of Complementary Quality Modeling approach, first, Model-2 is derived from Model-1 by appending new features (a.k.a operationalizations) as the extensions of qualitative quality modeling concerns. However, it is obvious that Model-2 falls short of responding to the needs for expressions required for feature interaction problems. As mentioned earlier, the features modeled in the context of Hardware Components feature group define certain restrictions on the quality concerns of a domain. As long as these restrictions are defined appropriately in terms of proper quality modeling abstractions (qualitative quality concerns in the case of Complementary Quality Modeling Approach), it is inevitable to define further exclude relationships between the functional features of Hardware Components feature group and different levels of qualitative quality modeling aspects. The reason why additional exclude relationships are required lies beneath the aim to get rid of feature interaction problems inherent in the model. At this point, it must be noted that Complementary Quality Modeling approach is said to be utilized for the sake of resolving these feature interaction problems.

In order to represent the feature interaction capability of Complementary Quality Modeling approach, relative FIPAP values of Model-2 and Model-3 with respect to Model-1 are calculated by means of equation (8) given in *Section 2.3.4.7*. Assuming En_1, En_2 and En_3 express the number of exclude relationships defined in the scope of Model-1, Model-2 and Model-3 respectively, En_1 turns out to be 5, En_2 is 7 and En_3 is 19. By utilizing these values in equation (8), relative FIPAP of Model-1 with respect to Model-2 is calculated as %40 whereas relative FIPAP of Model-1 with respect to Model-3 comes out to be %280. This is an indication of the fact that the Complementary Quality Modeling approach drastically enhances the feature interaction problem avoidance capability of the initial model. Besides, the proposed approach is said to be much more effective than Feature Oriented NFR analysis on the way to reveal further exclusion relationships implicitly included in qualitative quality modeling aspects.

5.2.2. Evaluation with respect to the Relation Between VF and ECR Criteria

As an extension on FIPAP criterion, ECR, originally defined by Mendonca et.al. [37], counts the requires relationships in addition to exclude relationships unlike FIPAP. VF is the indicator for the variability capability of a feature model. In the scope of evaluation of MMLS feature model extended with Complementary Quality Modeling approach, ECR is utilized for the comparative assessment of each model concerning its trade-off relationship with VF.

The trade-off relationship between the respective values of ECR and VF is based on usage density of structural and implicit interdependencies with respect to each other. Structural interdependencies play the role of enhancing the variability of a model whereas implicit ones are needed to keep this growth under control. Therefore a comparative increase in the usage of structural interdependencies leads to an increase in VF with a decrease in ECR, whereas an increase in implicit interdependencies causes an increase in ECR with a decrease in VF.

In order to manage this trade-off relationship efficiently, there should exist a balance between the respective values of ECR and VF sustained by a two way feedback. Sustainability of this balance is necessary for a model, especially if controlled growth in variability is of concern. An enhancement in the variability of a feature model is desired which can be easily realized by appending new features to the model. However, without paying attention to the interplay between the features, the feature model can easily slip out of control in terms of its ability to represent more realistic products. What is implied by more realistic products is that the combination of features included in a product should not obstruct overall functionality of the system due to feature interaction problems. Furthermore, some features need the presence of one another, therefore more realistic products should definitely include all these probable feature pairs possessing such kinds of require or exclude relationships.

On the way to achieve the goal of ensuring more realistic products from a feature model, a decrease in VF is required to be compensated by an increase in ECR.

In the light of these discussions, the parameters calculated for evaluation of VF and ECR are presented in Table 5.1 below, with respect to each model. A decremental trend in VF is observed as the Model-1 is evolved to Model-3 through Model-2. As for respective ECR values,Model-2 has its VF value decreased but can not benefit from this decrease in return for an increase in the definition of extra constraints. On the contrary, Model-3 compensates for this shortage caused by Model-2, by providing an enhancement in discovering additionally required relationships between features and QAs which are originated from quality aspects. Model-3 efficiently manages the trade-off relationship between VF and ECR by recovering the lack of feature interaction problem exploration capability of Model-2. In brief, Model-3 achieves the demand for sustaining feature relationships in required levels on the way to acquire more realistic products.

Table 5.1: Calculated Parameters of VF and ECR Criteria as per Model

Performance Metric	Model-1	Model-2	Model-3
Number of Products **[NP_T]**	$12 * 10^6$	$519 * 10^6$	$47 * 10^6$
Number of Products (without interdependencies) [NP_{WOI}]	2^{42}	2^{54}	2^{54}
Variability Factor (VF) [NP_T/NP_{WOI}]	$2.6 * 10^{-6}$	$0.03 * 10^{-6}$	$0.003 * 10^{-6}$
Number of Features involved in cross-tree constraints [NF_{CTC}]	25	28	38
Total Number of Features [NF_T]	42	54	54
Extra Constraint Representativeness (ECR) [NF_{CTC}/ NF_T]	0.6	0.5	0.7

5.2.3. Evaluation in terms of Commonality Criterion

As mentioned in *Section 2.3.4.3*, commonality is a substantial indicator of representativeness capability of any feature in terms of its contribution to possible product configurations. In other words, as the number of products in which a specific feature takes part increases, the respective commonality value of that feature increases as well.

The commonality values of all selectable features (i.e. leaf nodes of the feature model) with respect to each of three models are presented in Table 5.2 below. Model 2 enhances Model 1 with the fundamental contribution of the present study, namely feature oriented analysis of NFR goals. Features common to Model 1 and Model 2 are almost identical, except for the System Management feature group due to the specific relations of these features with the NFR goals in the analyzed domain.

Table 5.2: Calculated Parameters of Commonality as per Model

Feature Group	Feature	Model-1	Model-2	Model-3
System Management	HW-based Authentication	0.20	0.20	0.20
	PW Update Reminder	0.40	0.40	0.40
	Save Master-key	0.40	0.40	0.40
	Log System Setup Information	0.67	0.80	0.80
	Log Operational Information	0.67	0.40	0.40
	Unit – Level	0.28	0.28	0.43
	Module – Level	0.73	0.73	0.57
System Setup	Set/Change Data Recording Frequency	0.52	0.52	0.51
	Set/Change Data Transmission Frequency	0.52	0.52	0.51
	1280 x 1024	0.53	0.53	0.56
	1024 x 768	0.53	0.53	0.56
	Set Sensor Parameters	0.46	0.46	0.59
System Maintenance	Start-up Test	0.50	0.50	0.51
	On-line Test	0.50	0.50	0.51
	Off-line Test	0.73	0.73	0.57
	System SW Update	0.50	0.50	0.50
	Database Clean-up	0.50	0.50	0.50
	USB	0.25	0.25	0.25
	DVD	0.25	0.25	0.25
	CD	0.25	0.25	0.25
Information Delivery	Graphical Statistics Interface	0.50	0.50	0.48
	Maintenance Interface	0.99	0.99	0.99
	Data Transmission Interface	0.52	0.52	0.50
	Print Textual Statistics	0.67	0.67	0.67
	Print Graphical Statistics	0.67	0.67	0.67
Peripheral Delivery	RS422	0.93	0.93	0.99
	RS232	0.46	0.46	0.58
	VGA	0.80	0.80	0.64
	USB Port	0.25	0.25	0.25
	Ethernet/LAN	0.25	0.25	0.25
	Serial Port	0.25	0.25	0.25
	Parallel Port	0.25	0.25	0.25

Table 5.2 (cont'd)

Hardware Components	**Wind Speed and Direction Sensor**	0.46	0.46	0.15
	Ambient Humidity and Temperature Sensor	0.56	0.56	0.64
	Air Pressure Sensor	0.56	0.56	0.62
	Monitor Keyboard Drawer	0.80	0.80	0.29
	LCD Display	0.21	0.21	0.72
	Inkjet Printer	0.50	0.50	0.50
	Laser Printer	0.50	0.50	0.50
	Type A SSD	0.33	0.33	0.20
	Type B SSD	0.33	0.33	0.20
	Type C HDD	0.33	0.33	0.60
NFR Goals	**Low Resource Utilization**	-	0.20	0.20
	Medium Resource Utilization	-	0.20	0.20
	High Resource Utilization	-	0.60	0.60
	Low Accuracy	-	0.33	0.46
	Medium Accuracy	-	0.33	0.36
	High Accuracy	-	0.33	0.18
	Low Availability	-	0.33	0.26
	Medium Availability	-	0.33	0.40
	High Availability	-	0.33	0.35
	Low Cost of Ownership	-	0.33	0.69
	Medium Cost of Ownership	-	0.33	0.24
	High Cost of Ownership	-	0.33	0.07
AVERAGE COMMONALITY OF THE MODEL		0.49	0.45	0.45

The commonality values of NFR goal levels possessing almost equal shares in Model-2 are changed as the model is evolved to Model-3. This change brings about dominancy in terms of commonality values for some NFR goal levels. Model-2 and Model-3 display identical resource utilization levels while Model-3 displays significantly more favourable values, especially in terms of low cost of ownership and medium availability and low accuracy; all as required by the user.

This is a significant indicator of general quality-based characteristics of products derived from Model-3 in terms of different levels of qualitative quality modeling aspects. In other words, product configurations derived from Model-3 have the tendency to have high level of Resource Utilization, low level of Accuracy, medium level of Availability with low level of Cost of Ownership. Due to the strong connections between these NFR goals and functional features (in Hardware Components feature group) of the model by means of implicit interdependencies, Model-3 owes these quality-based characteristics to the properties of functional features (a.k.a. quantitative QAs) elicited in the scope of Complementary Quality Modeling approach. It is obvious that these quality-based characteristics are tentative and definitely dependent on the attributes of functional features.

Besides, the changes in commonality values of NFR goal levels from equal shares to dominancy in some levels shows that the integrability of NFR goal levels (qualitative quality modeling concerns) with the rest of the feature model is enhanced, as Model-2 is evolved to Model-3.

5.2.4. Evaluation in terms of DoO Criterion

Conceptually similar to the assessments on commonality values, DoO measure is utilized to reveal how competent Complementary Quality Modeling approach is in terms of integrability of its variability assets with the initial functional feature model. The level of integrability for a sub-tree (i.e. a feature group) in the feature model is proportional with the level of dependency with regards to feature selections to be realized during AE phase. Lower DoO for a specific feature group means higher degree of its integrability and dependency to other parts of the feature model.

One of the primary goals of Complementary Quality Modeling approach is to integrate elicited qualitative quality modeling aspects (NFR goals) to the rest of feature model. In order to assess the Complementary Quality Modeling approach regarding this concern, DoO values calculated separately for each NFR goal in Model-2 and Model-3 are needed to be compared.

Additionally, since integrability of NFR goals to the rest of the feature model is sustained by means of interdependencies with the functional features included in Hardware Components feature group, similar comparison is needed to be realized between DoO values of Hardware Components feature group in Model-2 and Model-3.

The DoO values with respect to Model-2 and Model-3, belonged to Hardware Components feature group and four basic NFR goals, namely Resource Utilization, Accuracy, Availability and Cost of Ownership are presented in Table 5.3 below. In this table, Model 1 is not considered as it does not refer to the NFR goals at all.

Table 5.3: Calculated Parameters of DoO for Model-2 and Model-3

Sub-tree	(Model-2)	(Model-3)
Resource Utilization	1.73×10^8	0.16×10^8
Accuracy	1.73×10^8	0.16×10^8
Availability	1.73×10^8	0.16×10^8
Cost of Ownership	1.73×10^8	0.16×10^8
Hardware Components	6.18×10^6	0.56×10^6

DoO values of NFR goals calculated with respect to Model-3 are lower than the values for Model-2. This obviously indicates that any selection realized among several levels of each NFR goal in Model-3 is more dependent on the rest of the feature model when compared with Model-2. This also means that decision of levels on each NFR goal in Model-3 has comparatively more effect on the selection of features from the rest of the feature model. Based on these evaluations, thanks to Complementary Quality Modeling approach, integrability of qualitative quality modeling aspects is enhanced as Model-2 is evolved to Model-3.

Additionally, DoO value of Hardware Components feature group for Model-3 is lower than the values for Model-2, similar to the case for NFR goals. This can be regarded as the contribution of Complementary Quality Modeling approach in terms of an enhancement on integrability of functional features to the whole model in parallel to the enhancement sustained for qualitative quality aspects as well. Therefore, the results can be regarded as an indication of convenience of the Complementary Quality Modeling approach in terms of its power to integrate qualitative and quantitative quality modeling aspects not only with each other but also with initial functional feature model.

5.2.5. Evaluation in terms of RPER Criterion

In the scope of MMLS domain, RPER (owing to its guidance in comparative assessment) is benefited for the evaluation of Complementary Quality Modeling approach regarding how well a variability modeling approach satisfies users and developers on the way to acquiring more permissible systems.

As referenced briefly in *Section 2.3.4.1*, due to difficulties in attaining its parameters (i.e. total number of acceptable product configurations), PER is utilized as part of RPER criterion under some restrictions. These restrictions set forth in *Section 2.3.4.1* are accepted beforehand for the evaluation of Complementary Quality Modeling approach in MMLS domain context.

In relation with those restrictions, it is meaningful to calculate RPER value with respect to the models possessing the same number of variability items, namely Model-2 and Model-3. If the models being compared differ in variability items (i.e. variants – leaf nodes of feature model), the results of RPER calculation may be misleading. This is due to the fact that without the presence of a common reference point for variability, relative value calculation of PER values may cause unexpected results. Hence, Model-1 is not utilized for RPER calculations, since as Model-1 is evolved to either of Model-2 or Model-3, the number of variability items is increased by operationalizations defined as the extensions of qualitative quality aspects (i.e. NFR goals).

Consequently, RPER of Model-2 with respect to Model-3 ($RPER_{2,3}$) comes out to be 0.09. This is an acceptable level of permissibility for Model-3 on account of possessing 11 times better accuracy in representing actual realizable systems.

CHAPTER 6

CONCLUSION

VM is one of the primary concerns having significant effects on all phases of SPL processes. Considerable effort has been devoted by the SPL community to develop various approaches on the way to go beyond the adversities laid by variability modeling practices. However, throughout the efforts expended for variability modeling on functionalities of domains, modeling of QAs derived from NFRs are neglected quite often.

On the way to meet the deficit of modeling quality-oriented concerns, as the beginning point of this thesis study, a comprehensive survey of the literature has been realized to identify the contributions and drawbacks of the state of the art approaches. This review has revealed the expressive power of feature-oriented QA modeling approaches, as feature oriented approaches have already proved their adequacy for functional concerns. Feature-oriented NFR Analysis seems to satisfy the need of modeling QAs originating from qualitative nature but lacks the need of modeling quality concerns possessing quantitative characteristics. On the other side, EFM approach remedies the lack of modeling quantitative quality aspects but lacks in responding the needs for modeling different levels of QA optionality at PL level. At this point, the proposed approach of Complementary Quality Modeling finds its meaning as it complements the two indispensable goals of QA modeling practices, namely modeling of qualitative and quantitative QAs. This integrated approach provides to domain designers the opportunity to exploit the quality knowledge– no matter it is either qualitative or quantitative – concealed in the specifications or requirements of any domain.

The proposed approach realizes the integration between quality concerns (in the form of levels as the extensions on different NFR goals) and functional features of a domain by serving the establishment of bridges in the form of dependency relationships. These relationships define available product configurations by means of direct effects on co-existence or non-existence of functional feature-quality concern pairs.

The most important contribution of Complementary Quality Modeling approach proposed and implemented in the scope of this study is the ability to perform trade-off selections among different types of basic quality concerns (namely NFR-goals) which not only provides further selectable operationalization alternatives but also ensures direct control over the available functional features by means of their elicited elementary attributes. In other words, management of variability among functional features is realized from the perspective of quality concerns by enhancing non-functional variability.

Different combinations of selected NFR-levels provide dynamism on the availability of functional features (i.e. features of Hardware Components feature group) at the time of product specification. During AE phase, as the levels of basic quality concerns (in terms of NFR-goals) are specified, the feature model becomes limited in terms of selectable functional features at the same time. This provides further feedback to product engineers for specification of more realistic product configurations dynamically.

Apart from contributions of Complementary Quality Modeling approach in non-functional variability management context as an alternative for state of the art QA modeling approaches; for the purpose of assessing Complementary Quality Modeling approach with regards to general PL benefits, PL metrics that are especially applicable on feature oriented models are collected from the literature. FAMA framework is utilized for calculating the parameters of these metrics particularly for three main states of the model throughout the implementation of the approach, namely Initial Functional Feature Model, Functional Feature Model extended with NFR goals and finally Functional Feature Model extended with Complementary Quality Modeling approach. The results for these three states are utilized for the comparative assessment of the proposed approach with respect to its contribution on Initial Functional Feature Model and Functional Feature Model extended solely by Qualitative Quality Modeling aspects.

Referring to the results obtained by means of the selected evaluation criteria, Complementary Quality Modeling approach obviously provides enhancement in exploring the feature interaction problems inherent in the domain of subject. This exploration is valuable in terms of revealing possible conflicts between quality concerns (in the form of levels as the projections of different types of NFR goals) and functional features. Besides, Complementary Quality Modeling approach succeeds the compensation for the subsidence of variability by ensuring feature relationships in satisfactory levels so as to acquire more realistic products.

Being one of the significant goals of the approach, Complementary Quality Modeling approach strive against integration of the elicited qualitative quality modeling aspects (NFR goals) to the rest of the feature model. As compared with Functional Feature Model extended solely by NFR goals, Complementary Quality Modeling approach provides better integrability of these qualitative quality concerns with functional features of the initial model. This integrability simply provides better means of comprehensiveness in modeling ability in terms of quality aspects. From the perspective of sustaining same type of integrability in a way that any selection realized among several levels of each NFR goal has strong dependency and impact on the selections to be realized among the rest of the feature model, Complementary Quality Modeling approach achieves its purpose.

It must be noted that, in the scope of MMLS feature modeling, the integration of elicited NFR goals are constituted with only Hardware Components feature group (as specific part of the whole feature model) instead of covering other parts of the model. Hardware Components feature group includes a large number of physical constraints. Regarding their capability to contribute to the definition of quality concerns, physical constraints are preferable for modeling. This is the basic reason why Hardware Components feature group is utilized for the modeling of Quantitative QAs. Even though it may be seen like a deficiency for the proof of concept, actually utilizing only Hardware Components is tentative and dependent on the nature of the domain of concern.

Since similar results with respect to the number of features, number of products and different relationships are expected to be observed in following the inclusion of extra artifacts applicable by the approach, the trends in evaluation are not expected to change. As feature groups other than Hardware Components are utilized for the modeling of Quantitative QAs, number of constraints established between features are increased, leading to an increase in FIPAP and DoO (thereby integrability of several feature groups to the model is enhanced); VF is decreased with an increase in ECR, leading to similar trend with the current MMLS case; total number of products is decreased, leading to a decrease in RPER value. It must be noted that, if other feature groups are included in modeling of QAs, all of the trends on evaluation would be similar and more favourable than the ones in the current study.

In the scope of this study, Feature Modeling of MMLS domain in addition to Qualitative QA and Complementary Quality Modeling practices are implementations on a small scale domain selected due to practicality and for the purposes of proof of concept. As for the generalizability of the approach, obviously, in a larger domain, more features are needed which come up with the need for more relationships to be defined. As of the outcome of implementation of Complementary Quality Modeling approach, more quantitative QAs would have to be defined together with more NFR goals. This would require support of an automated mechanism to provide guidance for matching the required levels of

NFR goals and values of quantitative QAs, both of which may be too numerous to be handled manually with the current approach.

Complementary Quality Modeling approach deals with the Domain Analysis phase of SPL process with an emphasis on QA modeling. As referenced in Figure 2.1, as a rule of thumb of DE phase, a feedback (often in the form of revised requirements) from AE phase is needed. Lack of support from AE phase causes threat on the validity of not only the feature model extended with Complementary Quality Modeling approach but also its outcomes in the form of Domain Model and developed software components. Therefore, as the components constituted from the feature model extended with Complementary Quality Modeling approach are utilized to form a final product, field knowledge in the form of experience of application engineers during software development or criticisms and demands from customer are needed to be taken into account and provided to domain experts as feedback in turn.

As the future steps to be traced on the way to improve and expand the benefits of the Complementary Quality Modeling approach, the interplay between the selectable levels of each elicited qualitative quality modeling aspects (NFR goals) can be utilized. This interplay is sustained by the trade-off selection availability among different levels of each NFR goal. With the help of this trade-off selection ability, some optimization operations can be realized on the affected functional feature variation points (i.e. Hardware Components feature group in MMLS domain example). For instance, users may desire to possess the optimum system with the lowest cost of ownership or regardless of other quality concerns stakeholders may desire to observe possible product configuration in which Resource Allocation is maximized. Complementary Quality Model provides reasoning to its users by presenting available product configuration possibilities based on such kind of optimization concerns.

REFERENCES

[1] Lee, K., Kang, K. C., Lee, J., "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering," *Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools, 2002, ISBN: 3-540-43483-6, pp. 62-77.*

[2] de Almeida, E. S., Alvaro, A., Lucredio, D., Garcia, V. C., de Lamos Meira, S. R., "A Survey on Software Reuse Processes," *International Conference on Information Reuse and Integration, 2005. IRI-2005 IEEE, pp. 66-71.*

[3] Li, H., Katwijk, J. W., "Issues Concerning Software Reuse-in-the-Large," *Proceedings of the 2nd International Conference on Systems Integration, 1992, ICSI'92, pp. 66-75.*

[4] Riva, C., Rosso, C. D., "Experiences with Software Product Family Evolution," *Software Architecture Group, Nokia Research Center, Proceedings of the 6th International Workshop on Principles of Software Evolution (IWPSE'03), 2003, pp. 161-169.*

[5] Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S., "Feature-Oriented Domain Analysis (FODA) Feasibility Study," *Technical Report CMU/SEI-90-TR21, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University (1990).*

[6] Griss, M. L., Favaro, J., d'Alessandro, M., "Integrating Feature Modeling with the RSEB," *Proceedings of 5th International Conference on Software Reuse, Victoria, BC,Canada (1998), pp. 76-85.*

[7] Gomaa, H., "Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures," *Addison Wesley Object-Oriented Technology Series, ISBN: 0201775956, 2005.*

[8] Supakkul, S., Oladimeji, E. A., Chung, L., "Toward Component Non-functional Interoperability Analysis: A UML-based and Goal-Oriented Approach," *2006 IEEE International Conference on Information Reuse and Integration, pp. 351-358, 2006.*

[9] Helferich, A., Herzwurm, G., Schockert, S., “Developing Portfolios of Enterprise Applications using Software Product Lines,” *Proceedings of the Conference on Component-Oriented Enterprise Applications (COEA 2005), Erfurt, Germany, 20 September 2005.*

[10] Peng, X., Lee, S., Zhao, W., “Feature-oriented Nonfunctional Requirement Analysis for Software Product Line,” *Journal of Computer Science and Technology, volume 24, Issue 2, pp. 319-338, March 2009.*

[11] Glinz, M., “On Non-Functional Requirements,” *RE’07, 15th IEEE International Requirements Engineering Conference, pp. 21-26, Oct. 2007.*

[12] Jacobson, I., Booch, G., and Rumbaugh, J., “The Unified Software Development Process,” *Reading, Mass.: Addison Wesley, 1999.*

[13] Benavides, D., Trinidad, P., Ruiz-cortés, A., “Automated Reasoning on Feature Models,” *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAISE 2005.*

[14] Kuusela, J., Savolainen, J., “Requirements Engineering for Product Families,” *Proceedings of the 22nd international conference on Software Engineering (ICSE), 2000, pp. 61–69.*

[15] Sinnema, M., Deelstra, S., Nijhuis, J., Bosch, J., “Modelling Dependencies in Product Families with COVAMOF,” *Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06), 2006, pp. 299-307.*

[16] Zhang, H., Jarzabek, S., Yang, B., “Quality Prediction and Assessment for Product Lines,” *Proc. of the 15th International Conference On Advanced Information Systems Engineering (CAiSE'03), 2003, LNCS 2681, Springer-Verlag, pp. 681-695.*

[17] González-Baixauli, B., Laguna, M. A., Crespo, Y., “Product Line Requirements based on Goals, Features and Use cases,” *International Workshop on Requirements Reuse in System Family Engineering (IWREQFAM), 2004, pp.4-7.*

[18] González-Baixauli, B., Leite, J., Mylopoulos, J., “Visual Variability Analysis for Goal Models,” *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE’04), 2004, pp. 198 – 207.*

[19] Chung, L., Nixon, B., Yu, E. and Mylopoulos, J., “Non-Functional Requirements in Software Engineering,” *Kluwer Academic Publishers, 2000.*

[20] Jarzabek, S., Yang, B., Yoeun, S., “Addressing quality attributes in domain analysis for product lines,“ *IEE Proceedings Software., Vol. 153, No. 2, April 2006.*

[21] Czarnecki, K., and Eisenecker, U.W., “Generative programming: methods, tools, and applications,” *Addison-Wesley, Reading, MA, 2000.*

[22] Etxeberria, L., Sagardui, G., Belategi L., “Modelling variation in quality attributes,” In *Proc. of the 1st Intl. Workshop on Variability Modelling of Software-intensive Systems, pages 51–59, 2007.*

[23] Bartholdt, J., Medak, M., Oberhauser, R., “Integrating Quality Modeling with Feature Modeling in Software Product Lines,” *ICSEA, pp.365-375, 2009 Fourth International Conference on Software Engineering Advances, 2009.*

[24] Pohl, K., Bockle, G.,Van Der Linden, F., “Software Product Line Engineering: Foundations, Principles and Techniques,” *Springer, 2005.*

[25] **V**an Der Linden, F., Schmid, K., Rommes, E., “Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering,” *Springer, 2007.*

[26] Chen, L., Babar, M.A., Ali, N., “Variability management in software product lines: A systematic review,” *in SPLC’09, pp. 81–90, San Francisco, CA, USA, 2009.*

[27] XFeature, http://www.pnp-software.com/XFeature/Home.html, last visited: 02 October 2010.

[28] Antkiewicz, M., Czarnecki, K., “FeaturePlugin: Feature Modeling Plug-in for Eclipse,” *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange, pp. 67-72, 2004.*

[29] Kästner, C., Thüm, T., Saake, G., Wielgorz, F., Apel, S., Feigenspan, J., Leich, T., "FeatureIDE: A Tool Framework for Feature-Oriented Software Development," *2009 IEEE 31st International Conference on Software Engineering, pp. 611-614, 2009.*

[30] Captain Feature, http://sourceforge.net/projects/captainfeature/, last visited: 02 October 2010.

[31] pure-systemsGmbH, "Technical White Paper, Variant Management with pure::variants," 2004.

[32] Fernandez-Amoros, D., Gil, R.H., Somolinos, J.C., "Inferring Information from Feature Diagrams to Product Line Economic Models," *ACM International Conference Proceeding Series Vol. 446, Proceedings of the 13th Internationsl Software Product Line Conference, pp. 41-50, 2009.*

[33] Benavides, D., Segura, S., Ruiz-Cortés, A., "Automated analysis of feature models 20 years later: A literature review," *Information Systems, 2010.*

[34] Clements, P.C., McGregor, J.D., Cohen, S.G., "The Structured Intuitive Model for Product Line Economics (SIMPLE)," *Technical Report CMU/SEI-2005-TR-003, 2005.*

[35] Metzger, A., Heymans, P., "Comparing Feature Diagram Examples Found in the Research Literature," *Technical Report, Software Systems Engineering University of Duisburg-Essen, 2007.*

[36] Czarnecki, K., Kim, P., "Cardinality-based feature modeling and constraints: A progress report," *In Proceedings of the International Workshop on Software Factories At OOPSLA 2005, 2005.*

[37] Mendonca, M., Wasowski, A., Czarnecki, K., Cowan, D., "Efficient compilation techniques for large scale feature models," *In Generative Programming and Component Engineering, 7th International Conference, GPCE , Proceedings, pages 13–22, 2008.*

[38] Kasikci, B.C., Bilgen, S., "Scalable modeling of software product line variability," *13th International Software Product Line Conference, SPLC 2009, San Francisco, August 2009.*

[39] Benavides, D., Segura, S., Trinidad, P., Ruiz-Cortés, A., “FAMA: Tooling a Framework for the Automated Analysis of Feature Models,” *First International Workshop on Variability Modeling of Software Intensive Systems (VAMOS), January 2007.*

[40] FAMA, http://www.lsi.us.es/~fama/ , last visited: 02 October 2010.

[41] Eclipse Platform, http://www.eclipse.org, last visited: 02 October 2010.

[42] Streitferdt, D., Riebisch, M., Philippow, I., “Details of formalized relations in feature models using ocl.,” In *Proceedings of 10th IEEE International Conference on Engineering of Computer-Based Systems (ECBS 2003), Huntsville, USA. IEEE Computer Society, pages 45-54, 2003.*

[43] Cysneiros, L.M., Leite, J.C.S.P., “Nonfunctional requirements: From elicitation to conceptual models,” *IEEE Trans. Software Eng., 2004, 30(5): 328-350.*

[44] ISO/IEC 9126-1:2001(E), *Software engineering — Product Quality — Part 1: Quality Model.*

[45] http://www.businessdictionary.com/definition/total-cost-of-ownership-TCO.html, last visited: 02 October 2010.