EMBEDDING QUALITY INTO SOFTWARE PRODUCT LINE VARIABILITY ARTIFACTS

Mworia Daniel, Nderu Lawrence and Kimwele Michael

Department of computing, Jommo Kenyatta University of Agriculture and Technology, Kenya,

ABSTRACT

The success of any software product line development project is closely tied to its domain variability management. Whereas a lot of effort has been put into functional variability management by the SPL community, non-functional variability is considered implicit. The result has been dissatisfaction among clients due to resultant poor quality systems. This work presents an integrated requirement specification template for quality and functional requirements at software product line variation points. The implementation of this approach at the analytical description phase increases the visibility of quality requirements obliging developers to implement them. The approach proposes the use of decision tree classification techniques to support the weaving of functional quality attributes at respective variation points. This work, therefore, promotes software product line variability management objectives by proposing new functional quality artifacts during requirements specification phase. The approach is illustrated with an exemplar mobile phone family data storage requirements case study.

KEYWORDS

Software Product Line Engineering, Functional and Non-functional requirements, Quality attributes, variability, integration and requirements specification.

1. INTRODUCTION

Non-functional requirements (NFRs) problems are grouped into definition problems, classification problems, and representation problems. Representation of NFRs is a big challenge owing to their fuzzy nature where depending on how we define an NFR; its representation on a software specification document can make it appear like a functional requirement creating even more confusion in requirements documentation.

Even though there are many continuing efforts to determine in which stage of software development to integrate NFRs, researchers agree that taking account of NFRs during the early phases of any software engineering can improve the quality and agility of software[1].

There are various ways in which NFRs can be represented depending on the reason for their use and phase of the software development project. Goal-oriented approaches have advanced well-defined approaches to model NFRs at the early stage of the requirement engineering process while at the architectural phase NFRs associated with particular components can be used to justify alternative designs [2].

Another very well-defined approach for representing NFRs is the textual representation which involves documenting requirements in software requirement specification (SRS) through the use of templates. The most widely used textual requirements representation methods are the natural language-based templates [3].

In any software development process, non-functional requirements (NFRs) analysis will yield performance requirements, business constraints, and non-functional properties or quality attributes (QAs). This work will focus on quality attributes requirements representation in software product line engineering (SPLE) where variability is critical and the operationalization of quality goals is closely interlaced with functional requirements.

In Software product line engineering (SPLE) requirements engineering activities are carried out in the early stages of domain analysis & engineering (DA&E). A product line is a set of products that share a common set of requirements, but also exhibit significant variability in requirements. In the requirements analysis stage, the requirements gathered in the previous stages are analyzed and further refined. The commonalities and variabilities can be identified either by using product line-specific techniques or other techniques such as feature-oriented domain analysis (FODA) and family-oriented abstraction, specification, and translation (FAST) [4].

SPLE exploits the similarities of the systems that belong to a product line and systematically handles the differences between them. Product line variability defines how product line applications may differ in terms of features, functionality, and quality requirements they fulfill. Like commonalities, product line variability is pre-planned by defining whether a given feature, functional or quality requirement is product line variability or not based on explicit decisions from all product management stakeholders [5] . Quality attribute variability can be due to functional variability causing indirect variation in qualities, and vice versa.

Most SPLE approaches typically cover the domain and application engineering processes but set aside one activity important to companies which is the analysis of non-functional properties (NFPs) or quality attributes and the evolution of SPL's artifacts. A large part of most SPL methodologies is the management of functional variability and the minor part of implementing quality variability is with annotations that are sometimes abandoned after a short period because of the lack of integration during the SPL development activities.

A literature review demonstrates the aspect of variability in quality attributes has been "neglected or ignored by most of the researchers and attention is mainly put in the functionality variability of the products. As observed in[3], most approaches to quality attribute incorporation in software product line development introduce the variability at the design level (e.g., within sequences diagrams) instead of modeling the variability of the quality attributes earlier on in the development process, such as the requirements analysis level or at the architectural level. Our approach addresses this gap by considering and integrating quality attributes at the domain requirements analysis and specification phase.

Feature Models are the most widely used variability language, that models variability through high-level features that are close to requirements specification. During feature model analysis it is important to consider quality attributes as part of the model variability alongside functional features to generate more than one solution, the variation points are made explicit and document the decision models with the knowledge necessary to ponder about the better solution for each product to be derived. This work, therefore, proposes an approach that will support the identification and integration of quality attributes with the functional features at respective variation point levels during the domain requirements analysis phase based on a higher-level abstraction of common features among variants.

The contribution of this paper is to provide support, applying domain analysis and variability management techniques, to the identification and representation of quality requirements in SPL development. This paper focuses on the analysis and specification of quality requirements alongside the functional requirements in the early stages of SPL development taking as input the domain requirement documents together with feature diagrams. The approach proposes the use of textual integrated requirements template to extract common functional and quality attribute requirements at the SPL variation point. Further, the approach extends the feature-based analysis of domain requirements by focusing on the product family variation points to generate common functional quality attributes among the product family variants which are then stored as aspectual components to promote reuse.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the proposed approach while Section 4 applies the conceptual approach to a case study. Finally, Section 5 summarizes the contributions of this work and outlines directions for further research.

2. RELATED WORK

A range of research works has been carried out that seeks to support the incorporation of NFRs in the software product line engineering (SPLE) process. Whereas there is no agreed-upon stage of integrating NFRs into the software development process efforts in the literature focus more on the solution space (design, architectural choice, evaluation, and testing) than the problem space(requirements elicitation and analysis) [6]. Some of the relevant approaches addressing this issue are presented below.

2.1. Quality attributes Integration based on Extension of UML models

To capture the variability of quality attributes using Model-Driven Development (MDD), [7] recommend annotating the base model employing extensions to the base modeling language. They add generic annotations related to quality attributes like performance to the UML model which represents the set of core reusable domain assets. The concrete UML annotations are based on UML profiles with stereotypes to achieve desired quality attributes modeling. However, annotations of the application base model prevent its reuse as well as that of derived quality attributes.

FeatuRSEB is a popular approach that combines FODA and the Reuse-Driven Software Engineering Business (RSEB) method. In FeatuRSEB UML-like notational constructs are used for creating Feature Diagrams, with explicit representation of variation points, and variants and explicit graphical representation for feature constraints and dependencies. Non-functional requirements are captured as feature constraints. Product Line Use Case modeling for System and Software engineering (PLUSS) is an approach that borrows from FeatuRSEB to combine Feature Diagrams and Use Cases. This approach makes explicit decomposition of the operator to compose a feature by introducing two new types of nodes; single adapters (represent XORdecomposition) and multiple adapters (OR decomposition). This approach however does not explicitly handle quality attributes.

2.2. Architectural based Quality Attributes integration Approaches

Extension of the feature model mechanisms from ATAM (Architecture Trade-off Analysis Method) can be used to represent quality attributes, their variability concerning optionality and levels, their influence on the quality of the functional, architectural, and implementation features (indirect variation). The extended feature model presents both functional and quality concerns as the fundamental elements used to capture the variability in subsequent phases of design and implementation.

At the architectural phase, existing works address quality attributes variability jointly with the variability of base applications. [16] propose the RiPLE-DE (RiSE Product Line Engineering - Design Engineering) process approach presenting the variability of quality attributes in feature diagrams and to derive the desired quality attributes the diagrams are enhanced with information of the base application (e.g., the system's response measure). The variation of attributes is presented in form of numerical values that will be used in the evaluation of the resulting architecture designing SPL architectures that involve the systematic transformation of functional requirements and quality.

Quality-driven Architecture Design and quality Analysis (QADA) is a method for incorporating attributes into software architectures, which do not however explicitly consider quality. Another approach in [8] suggests the influence of each feature on a non-functional property be predicted before generating the configurations. Their approach however focuses on predicting the effects of the features on individual applications instead of focusing on recurrent quality attributes at the domain engineering phase to promote reuse.

2.3. Goal Oriented NFR Integration Approaches

[9] conclude that there is an association between software product lines and goal analysis and thus one can use goal-driven requirements approaches for feature specification. Goal analysis modeling can support the auto-generation of feature models in SPLE. In the SPLE paradigm, an integrated modeling framework (F-SIG, Feature-Softgoal Interdependency Graph) extends the feature modeling with concepts of goal-oriented analysis. This goal-oriented analysis is aimed at letting developers capture the design rationale of inter-dependencies between variant features and quality attributes during the design of product line architecture, and evaluate the impact of variant features selected for a target system.

The goal-driven and Chung's NFR framework approach has been widely used by researchers to integrate NFRs into the software development process. Whereas functional requirements are considered as hard goals, non-functional requirements are presented as soft goals in the analysis specification process. The correlation is shown as a directed graph where the nodes are hard goals, the target nodes are soft goals and the edges are represented by the + or – characters However software developers pay more attention to the functional needs of software and NFRs such as performance, usability, reliability, and security are usually handled later in an ad-hoc manner mainly during the system testing phase [10].

NFRs can be essential in all aspects of Software Product Line (SPL) like in situations where a requirement may cut across all product lines and the variation exists in the contextual application. [10] recommended extending the Product Line Use Case modeling for System and Software engineering (PLUSS) to include other NFRs other than the performance NFRs only by use of discrete values to express the degree of satisfice-ability and for security NFR represent the levels of data protection as outlined in the NIST standard. This approach however focuses on how single NFRs can be evaluated for satisfiability during product testing.

[11] also proposes an approach of modeling quality attributes with the variability of the base application based on domain experts' judgments using the Analytic Hierarchical Process (AHP). This captured quality knowledge of domain experts is used for the quality-aware product. Any

functionality that affects quality attributes is referred to as a contributor but does not explicitly deal with the quality attributes.

[12] advanced another interesting approach known as Concern-Oriented Reuse (CORE), a general-purpose software development that leverages the strength of Model-Driven Engineering (MDE), Component-Based Software Engineering (CBSE), SPL, feature-oriented and aspect-oriented software development, and goal modeling to promote reuse. This approach entails encapsulating all software functional and non-functional Requirements in reusable units called concerns. As much as they do not explicitly deal with quality attributes, the encapsulation of concerns is what our proposed approach recommends. The other difference with our work is the fact that they model the variability of the component interfaces and not the integration of functional and quality attribute concerns like our proposal suggests.

2.4. Domain Requirements Analysis and Specification

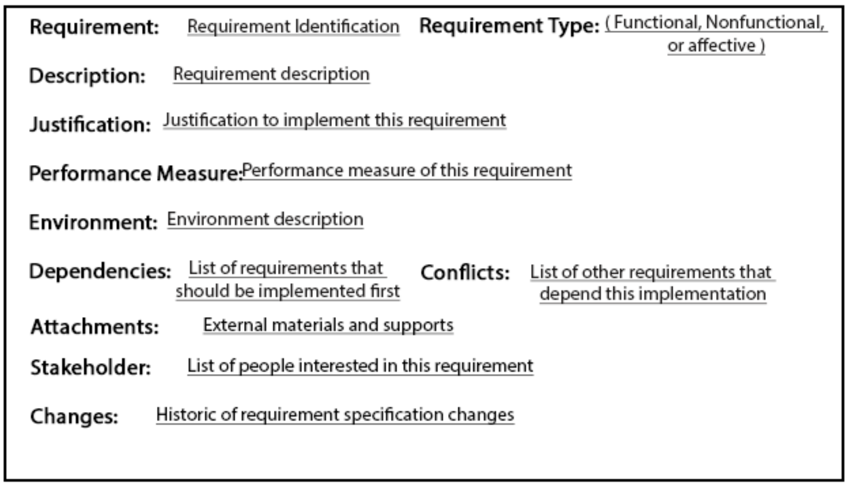
Our paper focuses on the textual representation of quality attributes alongside functional requirements in the software product line to support documentation and subsequent phases of development. We, therefore, mention related works in the line of textual analysis and representation of quality attributes alongside FRs both in SPLE and single-system development approaches.

According to [13] the most common approaches for analysis and specification for software product lines can be categorized as product-based specification, where the features of each product are specified one by one, and feature-based specification, where individual features are specified without links to any other features. There is also the family-based specification approach where specification can be written for all the features of the product line with variable parts for individual features. Our approach to SPLE specification is similar to the family-based specification with variable parts for individual features presented in a text-based specification method.

Whereas [14] note that software product lines do not have a de facto standard for requirements analysis and specification there have been several attempts that promote connecting goal-oriented approaches with this task. [9] observe that feature modeling is the core of software product line engineering and a de facto standard in modeling variability in SPL.

Extended feature models can address the representation of domain Quality attributes (such as performance, availability, security, or safety) including their variation dimensions. This work extends this approach by considering the quality attributes variability alongside the functional variability at variation points. Existing requirements documentation methods separate functional and quality attribute requirements whereas at the variation point there could be common variation to all possible family members that could be integrated and documented together as aspectual components for easier reuse.

Volere Requirements Specification Template is a well-established method for recording requirements in a structured way. The method supports the recording of user goals and requirements in the template according to their rationale, associated stakeholder, priority, and contextual details. There are different templates for specific NFRs like usability, maintainability security among others in the Volere documentation.



Requirement: Requirement Identification **Requirement Type:** (Functional, Nonfunctional, or affective)

Description: Requirement description

Justification: Justification to implement this requirement

Performance Measure: Performance measure of this requirement

Environment: Environment description

Dependencies: List of requirements that should be implemented first **Conflicts:** List of other requirements that depend this implementation

Attachments: External materials and supports

Stakeholder: List of people interested in this requirement

Changes: Historic of requirement specification changes

Figure 1. Volere Requirements Specification Template

The Volere Requirements Specification Template documentation inspired several other works including [15] and [16]. A problem with the usage of such templates is that they are useful only when a single person is responsible for managing them. However, in a project where many people are working simultaneously, this can lead to inconsistent, contradicting, and omitted requirements, and a need for a complex requirements management tool.

Apart from detailed tabular templates and models, several research works provide boilerplates (reusable sentences); a term referring to limited vocabulary sentences having specific placeholders to be completed to obtain semi-formal requirement sentences. [17]have presented an elicitation methodology by the use of their Non-functional Requirements Templates (NoRTs), which focuses on using generic statements(having a core and optional parts) that become defined NFRs after adding required information. EARS approach provides a simple boilerplate for requirement templates that can be used for non-functional requirements as well.

18] use natural language processing techniques for the identification of NFRs from requirements documents. The approach uses a language model and popular keywords for the identification of NFRs. This work suffers from the limitation of the lexicon or keywords as most NFRs are domain-dependent.

There have been different proposals for templates to support textual use case descriptions of Software Product Lines where fine-grained variation could be specified at the end of the SPL use cases with a template consisting of the following elements; name, type, line of the use case (the target of the variation), and description.

Another textual use case template found in [19] aimed at specifying the variation points through OPT and ALT tags where any text fragment of the textual use case description may be variant is explicitly marked by pairs of the XML-like tags <variant> and </variant>. [20] proposed a simpler tag notation where the tags are used only for marking variation points in use case scenarios of SPL. Each tag is expanded in a section called “Variations” and is mapped to the Orthogonal Variability Model (OVM).

[5] further, observe organizations can also use their specification templates or some standardized Software Requirements Specification (SRS) document structures to specify product line requirements. To capture the integrated quality attribute requirements at variation points we

propose to use our specification templates for documenting each variation point based on structured document templates such as extensible markup language (XML) which allow the hierarchical representation of common and variable requirements.

It is clear from the literature review that existing SPLE specification models mainly focus on feature models, use cases, and domain-specific requirements specification languages. These approaches represent functional and non-functional requirements in separate documents and diagrams but our proposed approach recommends integrated specification documents based on structured document templates such as extensible markup language (XML). The aspectual component development of the extracted functional quality requirement concern and the XML documentation can be handled using existing techniques in SPLE research. The research works included in this section can be summarized as follows:

a) Existing models to integrating quality attributes into the SPL development process do it more in the solution space (design, architectural choice, evaluation, and testing) than the problem space(requirements elicitation and analysis).
b) All SPLE approaches discussed in related work above support analysis of quality attributes in respect to the evaluation of achievement degree of non-functional property(NFP) in the final product but do not address the variation analysis of the quality attributes at the product family variation points.
c) Most of the text-based tabular templates represent quality attributes as independent elements of the requirements process. The need for NFRs' relationship with specific functional requirements is not fulfilled by most of these efforts.

This work, therefore, focuses on textual extraction and integrated representation of functional quality attributes at respective variation points during the domain requirements analysis phase by use of suitable decision tree class-attribute classification method. The Functional quality attributes can then be included in requirements documents to achieve traceability and incorporation throughout the development process.

3. PROPOSED APPROACH

Software Product Line (SPL) software development methodology is being adopted by many companies as opposed to single-systems software development. Variability modeling as a key domain engineering activity is continuously changing but tends to ignore the analysis of non-functional properties (NFPs) or quality attributes in the evolution of SPL’s artifacts[7]) A few organizations that attempt to implement NFP variability do so with annotations that are sometimes abandoned after a short period because of the lack of integration among the SPL activities.

In SPL development variability exists at different levels of abstraction, including requirements variability (mainly feature-based), architecture variability (mainly component-based), and implementation variability (mainly code-based). In most modern software systems variability can also be classified as variability in functional behavior, variability in non-functional system properties, and fault-based variability. This work focuses on requirements variability and possible integrated specification of functional and quality requirements in the early phases of software product line development.

From domain space knowledge & stakeholders requirements documents, in the feature-oriented analysis phase, we can extract common functional quality attributes among the variants and use hierarchical classification algorithms to extract common functional quality requirements and map them to respective variation points. The steps in the integrated specification of requirements using the decision tree classification method at the SPL variation points are as follows.

3.1. Identification of Various Point of Interest from an SPL Feature Model

User requirements and domain knowledge are subjected to Feature-Oriented Domain Analysis (FODA) to produce a feature model that captures commonalities and variability's of a product-line system during early development stages in form of a tree-based feature diagram. Functional features are decomposed into more fine-grained features that are mandatory, optional, or alternative, and optional features specify variability.

Product family variability is where a feature can have alternative implementations of variant implementations, which can be chosen to create different products. A variation point is each point in the software where different variant implementations from a variant population can be chosen from. Characteristics of a product that can be changed to produce a different product are called variation points. In terms of realization technique, a variation point can be the point where a class is chosen to be used or where code fragments are chosen to be run. Once you identify the related variant features of the product family in the graph a variation point can be marked with every set of related features [21].

Since a product family variability is occasioned by some stakeholder need in terms of system property or functions, the variation point could also yield members who also present common and variable requirements limited by the domain scope. Once the variation point is identified and labeled we propose an alternative template to specify functional, non-functional, and quality attributes for that variation point.

3.2. Analysis of requirements at the variation point

One way to incorporate the non-functional requirements early in the development of SPLs is to consider them at the variation point where common and variable features among the different variants can be analyzed to identify functional, non-functional, and quality requirements. There are common quality factors that are associated with functional requirements in each domain such as security for banking systems, reliability for embedded systems, and usability in general for most of the applications. These common quality factors in a family of software products when combined with functional requirements form functional quality requirements (FQAs).

At each variation point, the core functional and non-functional requirements of the family members can be identified and analyzed according to a structured specification template. Whereas non-functional requirements can be classified as performance, quality and constraints, our focus is on quality requirements because of their close relationship with functional requirements, especially in their operationalization. We, therefore, propose weaving certain quality factors with functional requirements as a way of embedding the quality factors at the variation points eventually supporting their satisfaction at the global level.

Considering that the major objective of software product line engineering is maximizing the commonalities (platform or architecture) while minimizing the cost of variations (i.e., of individual products) to facilitating reuse in a predictive manner this work contributes to this endeavor through further extraction of common functional quality attributes (FQAs) at each variation point. Whereas Feature-Oriented Domain Analysis (FODA) is a popular tool for variability identification it can be reinforced with a well-structured Natural Language (NL) requirements specification document as advocated in this and previous attempts [22].

In the Software product line paradigm, all requirements are captured in form of a feature model which is the base model or SPL domain knowledge space from which variant products are generated. Assuming that a domain knowledge space can be built using hierarchical decision tree classification techniques, all domain requirements can be specified, the variation points can be labeled as classes at respective nodes, and store variability concerns for that branch. This variability concerns could be the special functional, non-functional, or quality attributes different from the previous family requirement dimensions

3.3. Integrated Specification of Functional and Quality Requirements at a Variation Point

As [3] observed there are situations where a non-functional requirement affects neither a single functional requirement nor the system as a whole but a specific set of functional requirements. Such a case requires unique variability specification templates that ensure explicit documentation and adequate explicit traceability. This work proposes a semi-formal approach using structured non-mathematical notations to organize information about functional quality requirements.

At the candidate variation point of a feature model, we identify a dominant functional concern and decompose it into sub-features that contribute to its realization. We also identify a core non-functional property (quality attribute) of the domain at the variation point and refine it into specific quality concerns for each possible family member. Quality attributes such as security, usability, and error handling can be mapped directly to functional components and thus referred to as functional quality attributes (FQAs). These FQAs are normally required by several applications in a product line and therefore specialized components can assure their satisfaction.

A hierarchical decision tree classification algorithm can then be used to extract common functional quality attributes (FQAs) at each variation point and embed them to the node for use among family members emanating from that node. For a new product in the family line system, analysts will map it into the base model based on its specific requirements. We, therefore, propose that functional and non-functional requirements of a candidate product be processed using appropriate classification methods to identify functional quality attributes that can be added to the existing set at the respective variation point for future use.

An integrated textual requirements analysis template as in Table 1 can generate possible functional and non-functional requirements at the variation point exposing common functional quality attributes which apply to all members of the software product family with a common base at that variation point. The output of the integrated requirements analysis template can be stored in the extensible markup language format to support compatibility with most platforms. The generated FQA artifacts can be developed into aspectual components and through a join, the relationship is stored at the variation point node of the domain feature model for reuse.

Table 1. Elements of proposed variation point integrated requirements template.

Variation point(Vp) Id	(description)
VpFunctional Requirements	(description)
VpQuality Requirements	(description)
VpQualityConcern	(description)
VpFunctional-QualityConcern	(description)

The integrated functional-quality requirement becomes a new artifact in the SPL domain requirement specification space to be used in the subsequent phases of software development including design decisions. This weaving of quality attributes in functional description activity will remind the developers to consider them in all decisions and subsequent phases of software development [24].

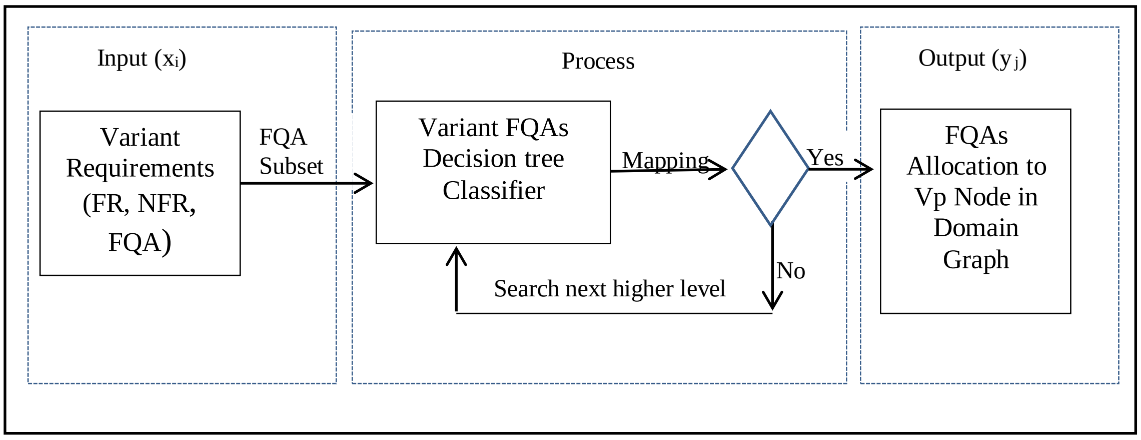

Input (x_i)
Variant Requirements (FR, NFR, FQA)
FQA Subset
Process
Variant FQAs Decision tree Classifier
Mapping
Yes
No
Search next higher level
Output (y_j)
FQAs Allocation to Vp Node in Domain Graph

Figure 2. The proposed approach for integrating quality concerns at SPL variation point

CASE STUDY

For practical demonstration of the proposed approach we present case study that consists of a simplified version of variability requirements from a mobile phone software product line family. Customizable software is necessary for a broad spectrum of domains (e.g., operating systems for diverse hardware) and hence our choice of mobile phone family data storage features programming.

Modern mobile phones are multifunctional and provide the ability to perform a wide range of actions beyond the common voice communication role. Common mobile phone features and utility functions include log in, call management, text messaging, storage, camera ringtones clock, and varying multimedia features. Among increasingly critical functions of a mobile phone is data storage which can be extended with flash memory card device and online backup. Phones as storage devices hold personal, organizational and even proprietary data.

Research findings consistently show that a significant portion of mobile phone users are concerned about security of their mobile device, its data, or its application against "casual" and unprofessional attack by children, spouses, friends, co-workers etc. Implementing this security feature for different members of the mobile phone family requires variability management in terms of functionality and quality attributes of the system. We focus on variability of the phone data protection and user privacy enforcement mechanisms as requirements that expose functional quality attributes at the variation points.

i) Identifying variation point dimensions

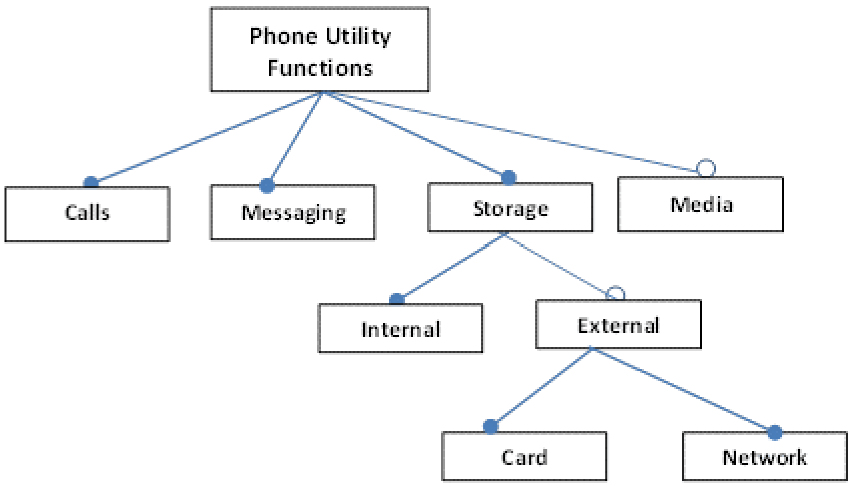


Figure 3: Mobile phone utility functions feature diagram

ii) Requirements analysis and specification at variation point

From Figure 3 we focus on storage features as a critical function in mobile phones today since they are being used as personal digital assistants (PDAs) for private and even corporate work. Variants in the mobile phone family line will present different abilities to satisfy that storage function. Assuming the following set of general user expectations from the phone family line expectations related to data storage:

Rq1.The phone shall have the capacity to store data
Rq2. The phone shall have the ability to extend the storage capacity
Rq3. The phone shall have the capacity to clear storage once full
Rq4. The phone may (optionally) permit the transfer of data to other devices
Rq5. The phone shall have the capacity to read different file formats
Rq6. The phone shall ensure the security of data
Rq7. The phone shall ensure user privacy

Rq6 and Rq7 are non-functional requirements and specific quality requirements which must be achieved by all variants to some level of satisfaction through different mechanisms. Addressing the satisfaction of the two quality requirements involves consideration of functional quality attributes at respective family tree variation points.

At the domain analysis activity, the requirements above will introduce further functional feature graph decomposition to bring out the different phone hardware mechanisms to operationalize them and possible limitations. The broad techniques of achieving the requirements are at the phone login, desktop, database, and external interface points as shown in Figure 4 with further variability among the possible solutions.

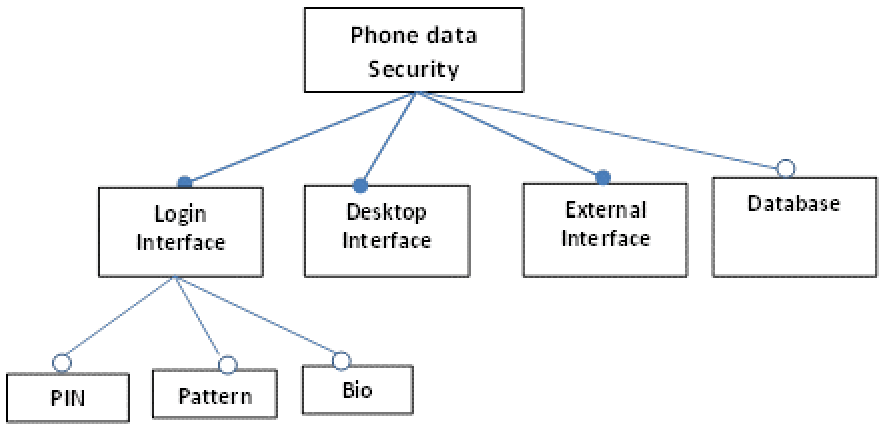


Figure 4. Mobile phone data security requirements feature diagram

iii) Integrated specification of Functional quality attribute requirements

Assuming we have three variants of the phone family that differ in their ability to satisfy the requirements Table 2 illustrates the possible scenarios.

Table 2: Functional quality achievement analysis matrix

Variant Type	**Ability to Satisfy**
Smart	Rq1,Rq2,Rq3,Rq4,Rq5
Evolving	Rq1,Rq2,Rq3,Rq4
Dumb	Rq1,Rq3

To support integrated specification of common functional quality requirements at the variation points we need to analyze the variants further concerning ability and quality attribute satisfaction mechanisms.

Looking at the security feature implementation capabilities for the different variants at different data access interface points a domain features function analysis template can generate the common functional quality requirements as shown in Table 3.

Table 3: Functional quality achievement analysis matrix

Phone Variant	**Login Interface**			**Desktop Interface**			**External Interface**		
	Pass	Bio	Patt	Pass	Bio	Patt	RW	H/w key	Enc
SMART	X	X	X	X		X	X	X	X
EVOLVE	X				X	X	X		X
DUMB	X				X				

Nb. Symbol (X) in the matrix denotes the variant that supports the associated security achievement mechanism, Pass (Password), Bio (Biometric) RW (Remote wipe), H/w (Hardware, Enc (Encryption), and Patt (pattern).

Table 3 presents an analysis template that indicates satisfaction of security and privacy quality requirements in the three variants phone data storage function happens in three dimensions (at login, Desktop, and External interfaces). However, the mechanisms of satisficing the quality requirements generate both common and variable mechanisms possible in the product line as follows:

At the Login security variation point, all three variants share the PIN authentication mechanism of access control, while some support pattern, biometrics, or both.

For Desktop security/privacy point all the three variants share auto- screen lock access control but differ in unlocking mechanism of PIN, pattern, biometrics, and key- combination.

For the External Storage security variation point, two variants share remote wipe and encryption capabilities but one has a hardware key and the other does not have the functionality.

The analysis above therefore generates three functional-quality requirements at the variation points as shown in Table 4.

Table 4. Functional quality achievement analysis matrix.

Variation Point	**Functional- Quality requirement**	**Specification ID**
Login Interface	Authenticate-PIN	VPlogin-Auth(PIN)
Desktop Interface	Display Lock –Auto/key lock	VPDesk-Lock(KEY)
External Interface	Encrypt	VPExt-Auth(encrpt)
External Interface	Remote wipe	VPExt-protect(Rw)

iv) Storage in the repository inform of XML aspectual component

The four common functional-qualities attributes (FQAs) for the three variants at different variation points can thus be developed separately as aspectual components to be attached to the common base architecture at respective join points defined by variation points. To make the requirements specification systematic and traceable the functional quality attributes can be stored in XML format in the repository together with the original SRS documents for future reuse.

This approach supports the architects and application engineers while generating new members or variants of the software product line family that are initially restricted by defined scope.

4. DISCUSSION AND CONCLUSIONS

In this paper, we suggested a practical approach for integrating functional quality requirements in SPL requirements documentation in an intuitive way. We have outlined steps in the process of analysis and integration and demonstrated the practicality of the proposed approach with a case study.

The proposed approach is based on domain feature model analysis and natural language textual representation, which is the most widely, used method in SPLE. Literature review shows a lot of variability analysis in functional dimensions while quality variability is considered implicit. Our approach, therefore, supports early consideration of quality attributes and their subsequent integration into the SPL documentation.

Since natural language and textual description of software requirements can be used to extract functional features and identification of variation points is a continuous activity in all phases including requirement gathering, this work attempted to extract quality attributes variations during analysis that can be represented alongside functional requirements owing to their means of operationalization. This work however is limited to incorporation and representation of quality attributes whose realization is based on a functional view of software.

One limitation in this work is the fact that it has not been tested in a complete product line architecture that specifies the rules on how the aspectual components will be connected as well as their relationships, interactions, and dependencies among them. For example, very elaborate security quality component implementation can negatively affect usability attributes and cost objectives. We, therefore, hope to investigate these scenarios in an industrial scope.

State-of-the-art solutions to modern-day problems demand automation which has not been accomplished in this work. To encourage adaptability of this approach we intend to develop a tool to manage automated extraction of functional quality attributes sets from software product line natural language documentation and map them to respective variation points using supervised decision tree classification algorithms.

REFERENCES

[1] R. R. Maiti (2016). Capturing, Eliciting, and Prioritizing (CEP) Non-Functional Requirements Metadata during the Early Stages of Agile Software Development. Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. (968) https://nsuworks.nova.edu/gscis_etd/968.

[2] L. Chung, B. A. Nixon, E. Yu,. & J. Mylopoulos (2012). Non-functional requirements in softwareengineering (Vol. 5). Springer Science & Business Media.

[3] G. Carvalho, F. Barros.and A. Sampaio A (2015). "NAT2TEST tool: From natural language requirements to test cases based on CSP." Software Engineering and Formal Methods. Springer, Cham, 2015. 283-290.

[4] J. M. Horcas ., M. Pinto & L. Fuentes (2019). Software Product Line Engineering: A Practical Experience. In 23rd International Systems and Software Product Line Conference - Volume A (SPLC '19), September 9–13, 2019, Paris, France. ACM, New York, NY, USA, 13 pages. https://doi.org/10. 1145/3336294.3336304

[5] A. Metzger, & K. Pohl, (2014). Software product line engineering and variability management: Achievements and challenges. FOSE. 10.1145/2593882.2593888

[6] J. M. Horcas (2018). WeaFQAs: A Software Product Line Approach for Customizing and Weaving efficient Functional Quality Attributes. A Doctoral Dissertation at the university of University of Malaga, Spain. Accessed online from http://orcid.org/0000-0002-7771-0575

[7] R. Tawhid, & D. C. Petriu, (2011) Automatic derivation of a product performance model from a software product line model," in 15th International Software Product Line Conference, ser. SPLC, 2011, pp. 80{89. 21

[8] N. Siegmund, M. Rosenm, Muller, C. Kastner, Giarrusso, P. G., S. Apel, and S. S. Kolesnikov, (2013). Scalable prediction of non-functional properties in software product lines: Footprint and memory consumption," Information & Software Technology, vol. 55, no. 3, pp. 491{507, 2013. [Online]. Available: https://doi.org/10.1016/j.infsof.2012.07.020 24, 26

[9] F. Q. Khan, S. Musa, & G. Tsaramirsis (2018). A novel requirements analysis approach in SPL based on collateral, KAOS and feature model. International Journal of Engineering & Technology, 7 (4.29) (2018) 104-108

[10] Nguyen, Q.L. (2009). Non-Functional Requirements analysis modeling for software product line. Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering, Washington, D.C., 56-61.

[11] G. Zhang, H. Ye, an& Y. Lin, (2014). Quality attribute modeling and quality aware product configuration in software product lines," Software Quality Journal, vol. 22, no. 3, pp. 365{401, Sep 2014. [Online]. Available: https://doi.org/10.1007/s11219-013-9197-z 20, 24, 26, 29, 78

[12] M. Schottle , O. Alam, J. Kienzle, & G. Mussbacher, (2016).On the modularization provided by concern-oriented reuse," in Companion Proceedings of the 15th International Conference on

Modularity, ser. MODULARITY Companion 2016. New York, NY, USA: ACM, 2016, pp. 184{189. [Online]. Available: http://doi.acm.org/10.1145/2892664.2892697 20, 25, 29, 78

[13] F. Q. Khan, , S. Musa, & G. Tsaramirsis (2018). A novel requirements analysis approach in SPL based on collateral, KAOS and feature model. International Journal of Engineering & echnology, 7 (4.29) (2018) 104-108

[14] S. Chimalakonda & D. H.Lee. (2016). On the Evolution of Software and Systems Product Line Standards. SIGSOFT Softw. Eng. Notes 41, 3 (June 2016), 27-30. DOI: http://dx.doi.org/10.1145/2934240.2934248

[15] C.Porter, E. Letier, & M. A. Sasse, (2014, August). Building a National E-Service using Sentire experience report on the use of Sentire: A volere-based requirements framework driven by calibrated personas and simulated user feedback. In 2014 IEEE 22nd International Requirements Engineering Conference(RE) (pp. 374-383). IEEE.

[16] M. F. A.Carvalhaes, A. F. d.Rocha , A. M. F.Vieira & T. M. G. d.Barbosa (2014). Affective Embedded Systems: a Requirement Engineering Approach. International Journal of Emerging Trends & Technology in Computer Science 8(2):70-75. DOI: 10.14445/22312803/IJCTT-V8P113

[17] S. Kopczyńska & J. Nawrocki, (2014, August). Using non-functional requirements templates for elicitation: A case study. In 2014 IEEE 4th International Workshop on Requirements Patterns (RePa) (pp. 47-54). IEEE.

[18] M. Younas, K.Wakil, D. N. Jawawi, M. A., Shah & A. Mustafa, (2019). An Automated Approach for Identification of Non-Functional Requirements using Word2Vec Model. International Journal of Advanced Computer Science and Applications(IJACSA), 10(8), 2019. http://dx.doi.org/10.14569/IJACSA.2019.0100871

[19] I. S. Santos, R.M. Andrade, and P.A. Neto (2015).Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment.. Journal of Software Engineering Research and Development (2015) 3:5 DOI 10.1186/s40411-015-0020-3

[20] W. Choi, S. Kang, H Choi, , J. Baik (2008) Automated generation of product use case scenarios in product line development. In: Proceedings of the International Conference on Computer and Information Technology. IEEE Computer Society,Washington, DC, USA

[21] González-Huerta, J., Insfran, E., Abrahão, S. and McGregor, J. D., 2012. Non-functional requirements in model-driven software product line engineering. In Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages - NFPinDSML ’12. New York, New York, USA: ACM Press, pp. 1–6

[22] A. Fantechi,. , S. Gnesi, & L. Semini, (2019) From Generic Requirements to Variability. Accessed on 5/12/2020 from http://ceur-ws.org/Vol-2376/NLP4RE19_paper16.pdf

[23] J. Jean-Marc (2012). Model-Driven Engineering for Software Product Lines. Review Article in ISRN Software Engineering Volume 2012, Article ID 670803, 24 pages doi:10.5402/2012/670803

[24] M. A. Gondal, N. A. Qureshi, H. Mukhtar, and H. Ahmed,. (2020). An Engineering Approach to Integrate Non-Functional Requirements (NFR) to Achieve High Quality Software Process. In Proceedings of the 22nd International Conference on Enterprise Information Systems - Volume 2: ICEIS, ISBN 978-989-758-423-7, pages 377-384. DOI: 10.5220/0009568503770384