ABSTRACT

Typically, products in a software product line differ by their functionality, and quality attributes are not intentionally varied. Why, how, and which quality attributes to vary has remained an open issue. A systematically conducted literature review on quality attribute variability is presented, where primary studies are selected by reading all content of full studies in Software Product Line Conference. The results indicate that the success of feature modeling influences the proposed approaches, different approaches suit specific quality attributes differently, and empirical evidence on industrial quality variability is lacking.

Categories and Subject Descriptors

D.2.13 [Software Engineering]: Reusable Software

Keywords

Quality attribute, Variability, Systematic literature review

1. INTRODUCTION

Software product lines have emerged as an important approach for efficiently developing varying software products [4]. In order to cope with this, software product line assets must explicitly manage and handle variability. Variability is the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context [35].

Quality attributes, such as performance or reliability, are often defined via a taxonomy of characteristics, e.g., [17]. Most studies in software product lines concentrate on functional variability. Thus, the state-of-the-art on quality attribute variability needs clarification. Further, there are reasons against intentional quality variability within industrial product lines: quality variability that affects architecture is difficult to be realized and is to be avoided [13]. Thus, the existence of industrial quality variability needs to be studied.

We present a systematically conducted literature review on quality attribute variability in software product lines. The method is adapted from the systematic literature review (SLR) guidelines [25]. The results conceptualize and elaborate a classification that includes dimensions for a rationale for and a means of capturing quality attribute variability. Further, specific quality attributes and empirical evidence on varying quality attributes are studied.

There exists one literature review on quality attribute variability in software product lines [8]; however, the focus was only on modeling aspects, and the method was not reported. Further, there exists a systematic literature review on quality evaluation of software product lines [29], but the analysis did not cover quality attribute variability.

2. METHOD

The research method was a systematically conducted literature review, i.e., an adaptation from the SLR guidelines [25]. The major modification was not to use search strings, but primary studies were selected by reading through all content in full studies in SPLC conferences. The reason for this was twofold. Firstly, we did not want to exclude any studies based on title and abstract only, since we knew several relevant studies, e.g. [30], that were not recognizable by their title or abstract. Secondly, it was impossible to construct a search string that would result in both relevant papers as well as give an amount of studies that could be subjected to reading all content through. As an example, one potential search string resulted in 122190 hits in IEEE database when subjecting the search on all content; and 408 hits when searching on metadata only and thus excluding studies based on abstract and title.

There were also minor modifications to [25]. Firstly, we did not assess the quality of the primary studies nor exclude any studies based on quality, mostly because the main focus of the primary studies was not on quality attribute variability. Secondly, we intentionally concentrated on qualitative instead of quantitative analysis. Quantitative analysis is beneficial for accumulating empirical evidence. In an immature topic, the primary studies tend to be qualitative or formative, proposing, e.g., a method. Therefore, the analysis and conclusions of the secondary study were qualitative, and generalizations were analytical [40].

The research questions were set as follows.

RQ1: What is the rationale for varying quality attributes in software product lines?
RQ2: How is quality attribute variability in software product lines captured?
RQ3: Which specific quality attributes are varying?
RQ4: What empirical evidence exists about varying quality attributes in industrial software product lines?
Table 1: Study selection

<table>
<thead>
<tr>
<th>Study Selection</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieving full studies from SPLC 2000 to SPLC 2010</td>
<td>221</td>
</tr>
<tr>
<td>Reading all content (title, abstract, body, figures)</td>
<td>21 match, 18 borderline studies</td>
</tr>
<tr>
<td>Reading the match and borderline studies again</td>
<td>29 match studies, selected for analysis</td>
</tr>
</tbody>
</table>

Table 1 describes how primary studies were selected. The first author did the manual reading and applied predefined inclusion and exclusion criteria in the selection:

**Inclusion**: says explicitly or uses an example/case where quality attributes vary in a software product line. Varying means that different products in a software product line have different levels of quality attributes.

**Exclusion**: contributes to quality variability by only using common feature definitions: a feature "is a user-visible aspect, quality, or characteristic" [19] or "is specified by a set of functional and quality requirements" [4]. These studies had too little contribution to be subjected to analysis.

**Exclusion**: does not address quality attributes observable via execution [3]. Varying quality attributes are argued to be observable via execution [6]. In fact, no concrete counterexamples were found in the manual reading.

Out of 221 initial studies, 29 studies were subjected to further analysis. The analysis was conducted in a bottom-up fashion, by first conceptualizing findings in the primary studies, and then elaborating categories from such conceptualizations. To assess researcher bias, a small-scale study selection replication was conducted by the second author. The first and the second agreed on the important match studies, but the first author accepted a slightly wider range of borderline studies. Also, no false negatives were found.

3. **RESULTS**

In the following, each section presents the results for one research question.

3.1 **Rationale for varying quality attributes**

Two reasons to vary quality attributes are identified.

The first rationale (Figure 1) is variation in the user and usage needs; some users prefer or require a different level of quality than others. Such a rationale may stem from different user, physical, social and business contexts [27], different geographical market segments [20], "high-end" and "low-end" segments [23, 24, 26], different privacy legislations and users' preferences [38, 15], dynamic usage changes, like user's hands becoming busy with other things [14], and even evolution of the data volumes and load over time [16].

The second rationale (Figure 1) is variation in hardware, resources, or functionality that directly affect the product quality attributes. Examples include different CPU and RAM capabilities that affect performance and memory consumption [39]; variation in network capacity, batteries running low, or devices running out of memory [14]; variation in product hardware and chipsets, which may cause the need to tune software to achieve optimal performance [22]; and variation in functional features, which may affect error rate and memory consumption [32].

The rationale also affects the strategy for achieving quality variability.

The first rationale, variation in the user and usage needs, does not cause any variants as such, but quality attribute variability has to be intentionally brought to the product line. This strategy is "differentiation" (Figure 1). Differentiation strategy needs a solution to create and derive variability in quality attributes. Examples of differentiation strategies include deriving feature configurations [27, 2], web personalization components [38], or service compositions [10, 34] to match a particular quality attribute requirement.

The second rationale, variation in hardware, resources and functionality, causes variation in quality attributes as such. Thus, there are two possible strategies (Figure 1). The first strategy is "impact management": just manage the quality attribute variation caused by the rationale, without building explicit mechanisms that try to affect resulting quality attributes. For example, variation in the game refresh time is caused by and evaluated against feature variability [7], without trying to actively select a particular feature configuration. The second strategy is "adaptation": the software is intentionally adapted to varying hardware, resources, or functionality, and explicit variation points or selection mechanisms are built for this purpose. For example, [14, 39] both derive a component composition that takes into account the varying amount of available memory.

3.2 **Capturing quality attribute variability**

Varying quality attributes are typically captured, i.e., modelled or specified in an explicit form. Two dimensions are identified (Figure 2).

Firstly, quality attributes can be captured as quality attribute features. Quality attributes are specified as feature-like entities, organized into trees, and variability is represented with mandatory, optional, and alternative relations. Quality attribute features can also have a refinement hierarchy: a quality feature can be composed into concerns, which can then be composed into scenarios [7]. As an example of this category, optional feature "Usability" has alternative child features "Usability for handicapped", "Usability for APT resident", and "Usability for VIPs" [27]. Quality attribute features are used in [27, 7, 10, 12, 11, 9, 20]. Not all studies call quality attribute features as "features", but the topology and relations resemble features very much.

Secondly, quality attributes can be captured by embedding them to other entities. Typically, quality is captured as a feature or component attribute, and variability is achieved by varying either attribute values or features and components. As an example, leaf features are attached with attributes that characterize memory consumption of that feature; composite features then specify how memory consumption is aggregated over its constituent child features [37]. Embedded approaches are used for features in [37, 2, 1], and for components in [39, 14, 38, 15].
Security variability is often addressed through functional variability: security is either realized with or affected by functionality. An example is [9], where security variability is achieved by varying functionality, like different access control methods. Another example is privacy variability in personalized web systems [38, 15], where personalization is characterized by privacy-violating functionality (e.g., "Cross-site tracking"). Further, security may have an order, like "low, medium, high" [1, 34], but even these levels are related to functionality (e.g., to encryption and dispatch service [34]).

Usability variability is mostly linked to functionality. "Usability for handicapped people" is enabled by extended elevator door time, whereas "Usability for APT residents" is enabled by cancelling elevator calls [27]. Alternatives "Speech-based UI" and "Text-based UI" imply different user interaction [14]. Usability is also captured as subjective user experience, like optional "Passenger comfort" [27] or "Attractive graphics" [7], which is then realized by functional features.

Other quality attributes, like reliability, correctness, accuracy, or error rate, are also mentioned [2, 20, 27, 30, 32].

Table 2 relates specific quality attributes to the dimensions described in Figure 2. Some categories are more heavily used for certain quality attributes. Memory consumption and performance lean more towards numeric and embedded approaches. In contrast, security and usability are often handled as discrete, separate features, perhaps since they are often treated through functionality. This may indicate that there is no "one-size-fits-all" approach for quality variability.

3.4 Cases, empirical evidence

The empirical evidence on industrial software product lines with varying quality attributes is scarce.

Firstly, two studies discuss direct empirical evidence on varying quality attributes. In the first study, three companies that have quality variability are described [30]. The varying quality attributes are observable via execution (security, performance, accuracy), and variation between wireless and wired communication causes quality variation [30]. Unfortunately, capturing or realizing quality variability is not covered. A second study seems to contain evidence on accuracy and coverage variability in the railway domain [24].

Secondly, there are studies that utilize an example and mention an industrial software product line. However, it is not known which parts of the examples are directly from the industrial cases, and which parts have been modified to highlight certain issues. Hence such examples cannot be considered as direct empirical evidence. The examples include a satellite communication software [37], an elevator control software [27], an enterprise system platform [16], and an automotive product line [5].

Finally, some studies use an example to illustrate or validate an approach. We do not consider such examples as empirical evidence on the existence of industrial quality variability.

4. DISCUSSION ON THE RESULTS

One observation is that the specific quality attribute, e.g., performance or security, needs to be considered, rather than quality attributes in general. Another observation is that the success of feature modeling seems to have a strong influence on the primary studies. We discuss these as follows.

Firstly, quality attributes are often proposed to be cap-
Table 2: Specific varying quality attributes and dimensions from Figure 2 against primary studies. Some primary studies mention quality variability only at a general level [12, 28, 29, 31, 36].

<table>
<thead>
<tr>
<th>Quality attributes</th>
<th>Memory consumption</th>
<th>Performance</th>
<th>Security</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality attribute features</td>
<td>-</td>
<td>[10, 27]</td>
<td>[9, 10]</td>
<td>[7, 20, 27]</td>
</tr>
<tr>
<td>Embedded to entities</td>
<td>[14, 37, 39]</td>
<td>2, 14, 21, 37, 39</td>
<td>[1, 15, 38]</td>
<td>[14]</td>
</tr>
<tr>
<td>Other</td>
<td>[5, 21, 24]</td>
<td>21, 23, 24, 34</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Possible variant space</td>
<td>Memory consumption</td>
<td>Performance</td>
<td>Security</td>
<td>Usability</td>
</tr>
<tr>
<td>Numeric metric</td>
<td>[14, 21, 37, 39]</td>
<td>14, 21, 37, 39</td>
<td>-</td>
<td>[14]</td>
</tr>
<tr>
<td>Discrete ordered levels</td>
<td>[5, 24]</td>
<td>2, 10, 23, 24, 34</td>
<td>[1, 10, 34]</td>
<td>[7]</td>
</tr>
<tr>
<td>Discrete characteristics</td>
<td>-</td>
<td>27</td>
<td>[9, 15, 38]</td>
<td>[7, 14, 20, 27]</td>
</tr>
<tr>
<td>Not enough information</td>
<td>[32]</td>
<td>6, 11, 16, 22, 26, 30, 32</td>
<td>[6, 11, 16, 30]</td>
<td>[6, 26]</td>
</tr>
</tbody>
</table>

tured as mandatory, optional or alternative quality features. It is yet unclear which specific quality attributes are amenable to be captured in such a hierarchy. Especially optionality may be problematic. Regardless of whether optional feature "Response time of status update < 3s" is selected, the status update function will always have a response time, and that response time may even accidentally be less than three seconds. In contrast, it is much easier to say when functionality is present or not present in the product.

Secondly, quality attributes may be transformed directly to functional features, e.g., security transformed to access control features [9]. Transformation from quality requirements to functionality is one architectural strategy [4]. Such transformation may be suitable for security or usability, but may be difficult for performance or resource consumption.

Thirdly, quality attributes are proposed to be embedded to functional features. This may be suitable for quality attributes that are specified in conjunction with functions, e.g., a feature "Call handling" with an attribute "response time". With such a feature, it is easy to specify and select a variant, but more difficult to realize the variability. If "Call handling" functionality is scattered among several components, it is difficult to implement varying "response time" values. In contrast, some studies embed quality attributes to components or services, thus localizing the variant realization in the scope of one component. However, such approaches need to characterize quality on a per-component basis, and to calculate the overall quality attribute value for a particular composition. As an example, how to embed usability as attributes into components, so that the overall product usability can be aggregated from these attributes?

Finally, the realization of quality attribute variability is often analyzed only against functional features. But quality attributes cannot be just byproducts of features. Since software architecture is the key in achieving many quality attributes [3, 4], realizing quality attribute variability requires explicit solutions in the product line architecture or components. For example, how to handle variability in data volumes [16] on the analysis of service performance [23, 24] by considering only the relation to functional features?

5. DISCUSSION ON THE METHOD

Traditional SLRs aim at completeness, objectiveness, replicability, and possibility to assess validity [25, 33]. We argue that our method fulfills all objectives but completeness similarly as traditional SLRs do.

The completeness of our method is different, since we intentionally set the scope to find all relevant studies from one forum instead of all possible forums. This scope causes a threat to external validity and generalizability. However, in certain aspects, the completeness of our method is deeper than in traditional SLRs. If a study is not recognizable by its title or abstract, or it uses different terminology, the search-based review will exclude the study. Thus, traditional systematic reviews aim at breadth in completeness, whereas our method aims at depth in completeness.

One systematic review uses, instead of search strings, manual reading of more than 100 journals [18]. Compared to our method, the manual reading in [18] excludes studies based on titles and abstracts. We did not exclude any studies based on titles and abstracts, but read all content through. Thus we had to limit manual reading to one forum.

Traditional SLRs search on metadata, and include and exclude studies based on titles, abstracts and keywords, before studying the content. However, this has several drawbacks. Firstly, abstracts and titles do not necessarily convey the contribution in such a way that search, inclusion, and exclusion can be based on them. If the main contribution of the study is different from the topic of the secondary study, even a well reported study may not be recognizable by its title and abstract. Most of the primary studies in our work were not recognizable by their title or abstract; this supports our decision to exclude papers based on full content. Secondly, studies in software engineering often use unestablished terms, and even more so for an immature topic such as quality attribute variability. Hence, it can be argued that search-based reviews are more suitable for analyzing topics that have an established name, like CMM and CMMI in [33]. Finally, if one wants to search for cases or examples, it may not possible to use only conceptual terms in search.

6. CONCLUSIONS

A systematically conducted literature review on quality attribute variability in software product lines was presented. The analysis covered a rationale for and a means of capturing quality variability, specific quality attributes, and empirical evidence.

Firstly, due to the diversity of quality attributes, there may not be "one-size-fits-all" approach for quality attribute variability. Specific quality attributes are treated differently in the primary studies. Secondly, the success of feature modeling is heavily affecting the proposed approaches. Questions still remain: which quality attributes can be captured as or analyzed against features, and how is architectural variability covered? Thirdly, there is a need for empirical descriptive accounts, e.g., by means of case studies [40],
instead of new methods. Empirical evidence is needed as follows: why and which quality attributes vary, what their variants are, and how they are realized in the architecture. Otherwise the research community cannot evaluate which approaches are needed and suitable.

As future work, we are extending the identification of primary studies with backward and forward references and studies from relevant authors. This will extend the scope to other forums, and thus shed light on the generalizability of our results.

7. REFERENCES


