Montagud Gregori, S.; Abrahao Gonzales, SM.; Insfrán Pelozo, CE. (2012). A systematic review of quality attributes and measures for software product lines. Software Quality Journal. 20(3-4):425-486. doi:10.1007/s11219-011-9146-7.

A Systematic Review of Quality Attributes and Measures for Software Product Lines

Sonia Montagud · Silvia Abrahão · Emilio Insfran

Abstract

It is widely accepted that software measures provide an appropriate mechanism for understanding, monitoring, controlling and predicting the quality of software development projects. In Software Product Lines (SPL), quality is even more important than in a single software product since, owing to systematic reuse, a fault or an inadequate design decision could be propagated to several products in the family. Over the last few years, a great number of quality attributes and measures for assessing the quality of SPL have been reported in literature. However, no studies summarizing the current knowledge about them exist.

This paper presents a systematic literature review with the objective of identifying and interpreting all the available studies from 1996 to date that present quality attributes and/or measures for SPL. These attributes and measures have been classified using a set of criteria that includes the life cycle phase in which the measures are applied; the corresponding quality characteristics; their support for specific SPL characteristics (e.g., variability, compositionality); the procedure used to validate the measures, etc. We found 165 measures related to 98 different quality attributes. The results of the review indicated that 92% of the measures evaluate attributes that are related to maintainability. In addition, 67% of the measures are used during the design phase of Domain Engineering, and 56% are applied to evaluate the product line architecture. However, only 25% of them have been empirically validated.

In conclusion, the results provide a global vision of the state of the research within this area in order to help researchers in detecting weaknesses, directing research efforts, and identifying new research lines. In particular, there is a need for new measures with which to evaluate both the quality of the artifacts produced during the entire SPL life cycle and other quality characteristics. There is also a need for more validation (both theoretical and empirical) of existing measures. In addition, our results may be useful as a reference guide for practitioners to assist them in the selection or the adaptation of existing measures for evaluating their software product lines.

Keywords *Software Product Lines · Quality · Measures · Quality Attributes · Systematic Literature Review*

Sonia Montagud

Department of Computer Science and Computation

Universitat Politècnica de Valencia

Camino de Vera, s/n, 46022, Valencia, Spain

Phone: +34 96 3877000, Fax: +34 96 3877359

smontagud@dsic.upv.es

Silvia Abrahão
Department of Computer Science and Computation
Universitat Politècnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
Phone: +34 96 3877000, Fax: +34 96 3877359
sabrahao@dsic.upv.es

Emilio Insfran
Department of Computer Science and Computation
Universitat Politècnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
Phone: +34 96 3877000, Fax: +34 96 3877359
einsfran@dsic.upv.es

1. Introduction

Software Product Line (SPL) engineering is a modern approach for developing a diversity of software products and software-intensive systems based on the underlying architecture of an organization product platform. It basically consists of two processes: *Domain Engineering*, in which a set of core assets are built, and *Application Engineering*, in which the core assets are used to derive a specific target product.

One of the most difficult tasks during product derivation is meeting the required quality attributes. A quality attribute is a measurable physical or abstract property of an artifact produced during the product line development. While quality is an important factor in the construction of individual software products, it becomes crucial in product line engineering since the quality of all the products of a family must be ensured. In addition, the development of SPLs has characteristics that distinguish it from the development of individual products. In particular, variability, reusability, commonality, and compositionality are key characteristics of this software production method that should be taken into account when evaluating quality.

Moreover, the life cycle model of software product line engineering is different from those of single product development [57]. In the Domain Engineering phase, the common architecture, the variabilities and commonalities of the SPL, along with the core assets that support these variabilities and commonalities are established, whilst in Application Engineering the products are built from the core assets. Also, in product lines, the architecture assumes a dual role [14]: there is the architecture for the product line as a whole and the architecture for each one of the products derived from the product line. The latter are produced from the former by exercising the built-in variation mechanisms to achieve product instances. Therefore, it is important to evaluate both architectures to ensure that (1) the product line architecture is robust and general enough to meet the product line scope and (2) the product architecture meets the specific quality attributes for a given product instance. Due to these issues, it is not possible to easily reuse quality assessment methods and techniques proposed for individual products.

In order to assess quality characteristics (both general characteristics and key characteristics for SPL), software measures[1] are a good means to understand, monitor, control, predict, and test software development and maintenance projects [10]. Over the last few years, a great number of measures for evaluating the quality of SPLs have been proposed. Several relevant quality attributes have also been identified. Unfortunately, despite the emergence of these measures, the state of industrial practice is still in its infancy. One possible reason for this might be the absence of information concerning the appropriateness and limitations of these measures for ensuring the quality of SPL. It is also difficult to establish the importance of new research achievements owing to the absence of a study that summarizes all the existing information related to quality attributes and measures for software product lines.

To collect this information, we present a systematic literature review. Systematic reviews are helpful in identifying, evaluating, and interpreting a research question, a study area, or phenomenon of interest in a thorough and unbiased manner [32]. They have also been successfully used in several areas of Software Engineering (e.g., software product lines [38] [13], requirements elicitation [16], usability evaluation [25], Web Engineering [36]). We have followed the guidelines proposed by Kitchenham [32] for performing systematic reviews in Software Engineering. To be effective, a systematic review must be driven by an overall goal. In this review, the main goal is the following: *identify and classify quality attributes and measures that have been proposed for assessing the quality of Software Product Lines.*

The remainder of this paper is organized as follows. Section 2 presents the background, including a discussion about the specific characteristics of SPLs and existing systematic literature

---

[1] In this study, we use the terms *measure* and *metric* as synonyms since the ISO/IEC 9126 standard [27] uses the term *metric* to refer to *the defined measurement method and the measurement scale* whereas the new standard for Software Product Quality Requirements and Evaluation (SQuaRE) [26] uses the term *measure* to refer to a variable to which a value is assigned as the result of measurement.

reviews in the field. Section 3 explains the research method employed. Section 4 reports the results of the review. Section 5 discusses the implications of our results for research and practice, along with the limitations of the review. Finally, Section 6 presents our conclusions and suggests areas for further research.

2. Background

In this section, we first provide a brief overview of software product lines engineering principles and their implications for evaluating quality. We then go on to discuss the existing systematic literature reviews published in the field of SPL.

2.1. Quality in Software Product Line Engineering

According to the Software Engineering Institute (SEI), software product lines are a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [14]. In this approach, the development of core assets is done by analyzing a target domain with the purpose of identifying common and variable features. The core asset thus realizes the commonality and variability among family members in the domain [53]. The resulting software will become unique through the addition or subtraction of features to/from a common architecture.

SPLs are used for the efficient and systematic development of software products. The expected benefits of this approach are to provide software systems with better quality and productivity, and lower development costs [14]. However, these benefits cannot be achieved if the reuse of software is not controlled and planned. A product that is part of a SPL is created based on the reuse of existing assets. Some of these assets are common to other products in the same product line and some are specific to individual products. An important issue in this context is, therefore, to capture and express variability and commonalities among different products.

SPL engineering makes a great impact on the quality of the resulting software [14]. To a large extent, a new application consists of matured and proven assets. This implies that the defect density of such products can be expected to be drastically lower than products that are all developed from scratch [57]. The production of software has historically focused on building individual product. However, with the arrival of the product line concept, new methods have appeared for software development. These new methods have a different development life cycle than that of traditional software [57], and the product lines have specific characteristics (e.g., variability, commonality, compositionality), thus signifying that existing methods with which to evaluate the quality of individual products cannot be easily applied to evaluate software product lines.

In recent years, several methods and techniques for evaluating quality in SPLs have been proposed (e.g., QRF [41], ETAM [30], HoPLAA [43]). Unfortunately, despite the emergence of these methods and techniques, the state of industrial practice for quality in SPLs needs to be updated since, in some cases, even the basic measurement principles are unknown. As a consequence, we believe that a study that summarizes and classifies the knowledge about quality attributes and measures for SPLs could be useful in allowing researchers and practitioners to improve their current practices.

2.2. Existing Systematic Reviews for Software Product Lines

We conducted several searches in the IEEEXplore, ACM Digital Library, and INSPEC digital libraries in order to ascertain the existence of any systematic review matching that presented in this paper. The following search string was used: *software AND product AND (line* OR famil*) AND (review* OR stud* OR systematic* OR evaluation* OR survey) AND (quality OR metric* OR measure* OR attribute*)*. However, we did not find any results.

Nevertheless, there are others systematic reviews published in the SPL field. In a previous work [38], we presented a systematic review in which we analyzed all the methods and techniques for assessing quality in software product lines. This study analyzed 39 research papers using 13 criteria (e.g. type of approach used for assessing quality, phase of the life cycle in which the quality assessment takes place, artifact(s) evaluated, mechanisms used to capture the quality attributes,

priority levels for quality attributes, impact analysis, validation procedure). The results indicated that many methods and techniques exist, but none of them considers all the characteristics which are desirable to ensure quality in SPLs. The study also indicated that several of the approaches used for evaluating the quality of SPLs rely on measures. However, this work was not specifically focused on the search and classification of measures for SPLs.

Despite the use of measures for assessing the quality of SPLs, there are very few works that discuss their usefulness. In general, the measures are presented individually. They are not integrated into a method for quality assessment that can be applied to different quality characteristics or applied throughout the whole life cycle of software product line development. For example, Etxeberria *et al.* [21] briefly mentioned the use of certain measures as a method for assessing the quality of product line architectures; however, there was no in depth discussion of the measures.

With regard to the research method used, we found other works that performed systematic literature reviews to identify and analyze measures in other areas of software engineering. For example, Calero *et al.* [12] presented a ranking of measures with which to evaluate the quality of Web information systems. The goal of this study was to provide an overview of the research in this area. The classification was carried out by using the Web Quality Model, which is a three-dimensional quality model for the Web. Gomez *et al.* [23] showed the trends in the software measurement field and the software process on which the measurement efforts have focused. They used a Software Measurement Ontology to classify and placed the amount of data in this field.

Other authors have also used systematic reviews as a means to analyze specific research questions in the SPL field [51], [13], [31], [42], [5], [44], [20]. For instance, Souza Filho *et al.* [51] analyzed domain design approaches through a systematic review, which may be useful in allowing companies to understand the current scenario and to choose a more suitable approach or adapt it to their needs. Chen *et al.* [13] conducted a systematic review to provide an overview of different aspects of the proposed approaches for managing variability in SPLs (e.g., the evolution of the research on developing variability management approaches, the key issues that have driven the evolution of the different variability management approaches). Khurum and Gorschek [31] conducted a systematic review analyzing the level of industrial application and/or empirical validation of the proposed solutions for the purpose of mapping maturity in terms of industrial application. They went on to evaluate the usability and usefulness of the proposed solutions.

Odia [42] performed a systematic review to summarize the existing evidence regarding testing techniques in SPLs. This study addressed research questions regarding the relationship between product lines and product; the steps needed to develop and execute testing in SPLs; the approaches that can be used to develop test cases; and the connections between software reuse and reusability of test cases.

Alves *et al.* [5] conducted a systematic review in requirements engineering for SPL. The results of this study showed that the adoption of the requirements engineering methods is not yet mature. The proposed approaches still have serious limitations in terms of rigor, credibility, and validity of their findings. A further major drawback is that it does not provide sufficient guidance for practitioners interested in adopting the proposed methods, thereby limiting its usability in industrial settings. In particular, there are very few commercial or open source tools available. Another remarkable result is that the proactive strategy is the most common adoption strategy suggested by the methods.

Finally, Pérez *et al.* [44], and Engström and Runeson [20] analyzed the state-of-the-art in testing. Pérez *et al.* [44] conducted a systematic review in order to identify experience reports and initiatives carried out in Software Engineering related to testing in SPL. The extracted information of each study is: the category of testing (e.g., Unit Testing, Integration Testing, Functional Testing, SPL Architecture Testing); how variability is dealt with; the testing technique used; if there is a tool or prototype to support it; and if the proposal was tested in an industrial or artificial setting. Engström and Runeson [20] conducted a systematic mapping to get an overview of existing research of product line testing. This included gathering identified challenges for testing SPL; recognizing the research forums SPL testing is published in; looking at which topics in testing SPL have been investigated and to what extent; and investigating what types of research are represented and to what extent. The conclusion of this study was that more validation is needed to provide a better foundation for testing in SPL.

3. Research method

A systematic review is a research method which is developed to obtain, evaluate, and interpret all the information that is related to a specific research question or area of interest. Its purpose is to provide an objective assessment of a research topic in a reliable, rigorous, and methodological manner, and it was for this reason that we selected a systematic review as a research method to conduct the study. Specifically, we followed the approach suggested by Kitchenham [32].

A systematic review involves several stages and activities (see Figure 1), which are briefly explained below:

1. Planning the review: the need for the review is identified, the research questions are specified, and the review protocol is defined.
2. Conducting the review: the primary studies are selected, the quality assessment used to include studies is defined, the data extraction and monitoring is performed, and the obtained data is synthesized.
3. Reporting the review: the dissemination mechanisms are specified, and the review report is presented.

Figure 1. Activities in the Systematic Literature Review Process

The activities concerning the planning and the conducting of our systematic review are described in the following subsections. The report of the review stage is presented in Section 4.

3.1. Planning the review

The objective of this systematic review is to obtain and analyze the quality attributes and measures proposed for assessing the quality of SPLs. The study will identify gaps and new areas of research for further investigation.

Following the systematic review methodology, we will first formulate the research questions, which are described as follows:

- What *quality attributes* have been proposed for assessing the quality of software product lines?
- What *measures* have been proposed for assessing the quality of software product lines and how are they used?

Kitchenham indicates that some guidelines recommend considering and framing research questions by following several criteria adapted to Software Engineering which have been collected from several other sources [32]. These are the following: population, intervention, comparison, outcomes and context. Population, in software engineering, could be any of the following: a specific software engineering role, a category of software engineering, an application area, or an industry group. Intervention is the software methodology, tool, technology or procedure that addresses a specific issue. Comparison is the software engineering methodology, tool, technology or procedure with which the intervention is being compared. Outcomes should relate to factors of importance to practitioners. Finally, the Context in Software Engineering is the context in which the comparison takes place: the participants taking part in the study being performed, and the tasks that are being performed. In our systematic review, these criteria are expressed as follows:

- **Population:** This corresponds to studies that propose quality attributes for SPL and/or propose measures for evaluating the quality of SPL.
- **Intervention:** This is the set of quality attributes and measures that are used for evaluating the quality characteristics of SPLs.
- **Comparison**: We do not compare the quality attributes and measures with other techniques or models. Our intention is to classify them based on specific criteria.
- **Outcomes**: To gather the attributes and measures proposed by the researchers for evaluating the quality of SPLs.
- **Context**: We are working in a research context in which there are experts in the domain.

3.1.1. Identification of data sources and search strategy

To obtain the primary studies, we consulted the digital libraries of the most relevant organizations in the Software Engineering community. We selected the following: IEEEXplore, ACM Digital Library, Science Direct, SpringerLink, and INSPEC.

The search covered the period from 1996 to July 2009. We chose 1996 because it was the first year in which a conference specifically dedicated to SPL was held. With regards to the search options of the digital libraries, we ensured that the search included magazines, journals, and conference proceedings.

We tested several search strings, and the following retrieved the greatest number of relevant papers: *((attribute* OR factor* OR propert* OR criterion OR criteria OR characteristic*) OR (metric* OR measur*)) AND (software OR engineering OR architectur*) AND ("product line*" OR "product famil*") AND (quality OR non-functional OR "no functional" OR assess* OR assur*).*

The search was conducted using the titles and abstracts of the articles. We also adapted the search string notation for each digital library, since they each use a different syntax. In order to assess the results of the search process, we compared the results obtained with a small sample of primary studies that had previously been identified as studies expected to appear in the results, this allowed us to ensure that the search process was able to find the relevant sample.

We additionally selected those conference proceedings and journals in which studies relevant to our domain had previously been published (see Table 1). These conference proceedings and journals were checked in order to determine whether or not they are indexed in the digital libraries for each year included in our study. We performed a manual search of them in order to:

- Complement the automated search to cover the cases in which the conference proceedings or journals did not appear in the digital libraries.
- Validate the search string in cases in which the conference proceedings or journals appeared in some digital libraries.

Table 1. Conferences selected for manual search

Software product line	Software quality	Software engineering
SPLC (Software Product Line Conferences), **PFE** (International Workshop on Product-Family Engineering) **IWSAPF** (International Workshop on Software Architectures for Product Families).	***ESEM*** (Empirical Software Engineering and Measurement) **ISESE** (International Symposium on Empirical Software Engineering) **MENSURA** (Int. Conference on Software Process and Product Measurement) **METRICS** (IEEE International Software Metrics Symposium) **QSIC** (Conference on Quality Software) **QoSA** (Quality of Software Architectures) **RefsQ** (Requirements Engineering: Foundation for Software Quality).	**ICSE** (International Conference on Software Engineering) **CAiSE** (Int. Conference on Advanced Information Systems Engineering) **MODELS** (Model-Driven Engineering Languages and Systems) **ECSA** (European Conference on Software Architecture).

To broaden the study as much as possible, we added grey literature to the review that did not appear in the digital libraries used. Specifically, we added primary studies that were known in advance, along with other studies that were found by using several combinations of the same search string in Google and Yahoo. However, the references included in the primary studies were not systematically followed up since most of the relevant conferences were already manually or automatically searched (see Table 1). Several systematic reviews conducted in the Software Engineering field (e.g., [5], [13], [44]) have also been left out or replaced in the secondary search by other practices. Nevertheless, the previously mentioned grey literature included six new primary studies that did not appear using the automatic search.

3.1.2. Selection criteria for primary studies

The search performed with the search string is a syntactic search, i.e., the result of the search is the set of papers in which the research string appears. However, on occasions, a given study may match the search string but the topic of the paper is different to the study area. As such syntactic search must be performed with a semantic check. This is done by reading the title and abstract of the papers in order to ensure that the collection is consistent with the research area under study. This selection is performed in an objective manner by applying the inclusion and exclusion criteria (shown in Table 2) whilst reading the titles and abstracts of the potential studies.

Table 2. Inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
▪ Papers which present measures for quality assessment in software product lines. ▪ Papers showing attributes that are desirable measures for assessing quality in software product lines.	▪ Papers not written in English. ▪ Papers proposing measures which are not directly related to the quality (e.g. cost models). ▪ Papers that propose measures but do not explain how they are used. ▪ Introductory papers for special issues, books, workshops or posters. ▪ Papers presenting measures for assessing other types of quality (process quality, quality of use, etc.).

However, titles and abstracts are not extensive and thus are not always clear indicators of what an article is about. For this reason, in some cases we scanned the full paper in order to ensure whether the data satisfied the inclusion and exclusion criteria.

3.1.3. Data extraction strategy

To extract the data from the set of selected primary studies, we divided the research questions into various criteria. Table 3 summarizes the data extraction criteria for classifying the collected quality

attributes and measures. Note that for each measure, we indicate the quality attribute it was measured by. For each quality attribute with a measure associated with it, we indicate the type of attribute (internal or external) and its domain (SPL relevant attribute or general attribute).

In the analysis of results section, we summarize the most significant results regarding the quality attributes. We also discuss those quality attributes considered to be most relevant for researchers, along with the number of measures proposed for measuring each one of them.

Table 3. Data extraction form

1. Quality characteristic evaluated
☐ Functional Suitability ☐ Reliability ☐ Performance efficiency ☐ Operability ☐ Security ☐ Compatibility ☐ Maintainability ☐ Transferability
2. Quality attribute evaluated by the measure
Name of attribute
•
2.1. Type of attribute
☐ Internal ☐ External
3. Type of measure
☐ Base Measure ☐ Derived Measure
4. Result of the measure
4.1. Type of evaluation
☐ Qualitative ☐ Quantitative ☐ Hybrid
4.2. Precision
☐ Exact ☐ Probabilistic
5. Phase of the life cycle in which the measure is applied
Domain Eng.: ☐ Requirements ☐ Design ☐ Test ☐ Realization Application Eng.: ☐ Requirements ☐ Design ☐ Test ☐ Realization ☐ Evolution
6. Artifact(s) used to calculate the measure
☐ Common Architecture ☐ Core Asset ☐ Product architecture ☐ Final product
7. Other characteristics
7.1 Compositionality
☐ Yes ☐ No
7.2. Variability/Commonality
☐ Yes ☐ No
8. Validation procedure
☐ Theoretical validation ☐ Empirical validation ☐ Not validated
8.1. Theoretical validation
☐ Property-based approach ☐ Measurement-theory approach
8.2. Empirical validation
☐ Case studies ☐ Surveys ☐ Controlled Experiments
9. Tool support
☐ Manual ☐ Automatic
10. Current usage
☐ Academic ☐ Industrial

Each data extraction criterion (with the possible options) is explained as follows:

Criterion 1. Quality characteristic evaluated. We classified each measure that was extracted from the selected papers according to the quality characteristics suggested in the ISO/IEC 25000 (SQuaRE) standard [26]. Thus, each measure was classified in a quality characteristic if it measures a quality attributed related with: *Functional Suitability, Reliability, Performance efficiency, Operability, Security, Compatibility, Maintainability,* and *Transferability*. For more information about these quality characteristics readers are referred to [26].

Criterion 2. Quality attribute evaluated by the measure. The measures measure quality attributes. This criterion indicates the quality attribute evaluated by the measures. It also indicates the *type of attribute*. The quality model defined in the ISO/EIC 9126-1 standard for product quality evaluation classifies quality attributes as external (visible on the system level), and internal (properties of subsystems and components). A measure is thus classified as being *Internal* if the quality attribute is measured at analysis or design time. If the measure can only be applied at runtime, or once the product has been developed, then it is classified as *External*.

Criterion 3. Type of Measure. According to the SQuaRE standard [26], there are two types of measure: base measure and derived measure. A base measure is defined in terms of an attribute and the method for quantifying it while a derived measure is defined as a function of two or more base measures. A measure is therefore classified as a *Base Measure* if it is functionally independent of other measures and as a *Derived Measure* if it uses other measures.

Criterion 4. Result of measure: We have divided the types of results into two categories: (1) those related to the type of evaluations (i.e., quantitative, qualitative or hybrid) and (2) those related to the precision with which the attribute is measured (i.e., exact, probabilistic):

- **Type of evaluation.** A measure can measure a quality attribute in a qualitative, quantitative, or hybrid manner. *Quantitative* evaluations are concerned with evaluating the attributes numerically, using continuous values (e.g., an attribute can be measured with continuous values between 0 and 1). *Qualitative* evaluations are those that indicate qualities (e.g., an attribute can be measured as high, medium or low). *Hybrid* evaluations are those that use both qualitative and quantitative evaluations.

- **Precision.** The measurement values obtained by applying the measure can be exact or probabilistic. The value is exact when the result of the measure is a numeric value (e.g. the sum of the number of classes). The value is probabilistic when the result of the measure is a percentage (e.g., the probability of improving the software usability).

Criterion 5. Phase of the life cycle in which the measure is applied. In this work, we have classified the measures by adopting the life cycle proposed by Van der Linden *et al.* [57]. According to this framework, the *Domain Engineering* phase contains four phases:

(1) *Domain Requirements Engineering* is the process of creating and managing requirements for the entire product line. A measure is therefore classified in this phase if it is defined for measuring the quality of domain requirements artifacts;
(2) *Domain Design* is the process of creating a common architecture for the SPL. A measure is classified in this phase if it is defined for assessing the quality of software artifacts related to the design of the whole product line and/or common architecture, taking into account the variability, the common parts, etc.;
(3) *Domain Realization* is the process of creating the core assets. A measure is classified in this phase if it is defined to evaluate the quality of core assets;
(4) *Domain Testing* is the process of testing the common architecture and core assets created. A measure is therefore classified in this phase if it measures a quality attribute related to the testing phase (for example, the quality of the proven cases).

The phases of the *Application Engineering* phase are:

(5) *Application Requirements Engineering*, which addresses the requirements of a particular product within the product line. A measure is thus classified in this phase if it measures quality attributes in the artifacts of the Application Requirements phase (e.g. use cases, scenarios);
(6) *Application Design*, which is a product architecture that is derived from the common architecture. A measure is therefore classified in this phase if it evaluates the quality of the product architecture;
(7) *Application Realization*, which addresses the implementation of products using the specific core assets and unique core assets. A measure is accordingly classified in this phase if it is defined to measure the quality of the implemented products;
(8) *Application Testing*, which addresses the testing of the implemented product. A measure is classified in this phase if it measures the quality attributes of a final product.

In addition to these phases, *we also include another phase* suggested by Bosch [9]:

(9) *Evolution*, which is driven by changes in the requirements of the products in the product line. A measure is thus classified in this phase if it evaluates whether the product line fits new requirements and some of the desired quality attributes.

Criterion 6. Artifact(s) used to calculate the measure. In order to discover which types of artifact are used to apply the measures, we have established the following classification: (1) *Common architecture*, if the measure measures the architecture of the entire product line created in the Domain Engineering phase; (2) *Assets*, if the measure measures a component (a large-grained reuse unit) of the product line; (3) *Product Architecture*, if the measure measures the architecture of a particular product; (4) the *Final Product*, if the measure measures the product that is produced in the Application Engineering phase.

Criterion 7. Other characteristics: Software product lines have their own characteristics that are specifically relevant to them or that are not present in other software development approaches. We therefore wished to discover which measures were defined by considering the following characteristics:

- **Compositionality.** The main idea of compositionality in SPL [47] is that the software platform of the product line is not a fully integrated software solution, but a set of components, architecture guidelines, principles as well as testing, documentation and use case support. Thus, the final products are composed, integrated and tested. If the measure takes into account the different artifacts and the compositionality among them, it is classified as *Yes* (e.g., measures that take into account the relationship among assets, the relationship between the architecture of the product line with the assets, etc.). If the measure evaluates only one artifact without taking into account issues related to others artifacts, it is classified as *No*.

- **Variability/Commonality.** In SPL, variability provides the required flexibility for product differentiation and diversification while and the commonality forms a common base. In [57] three types of variability are presented: Commonality, Variability and Product-specific. Commonality is when a characteristic (functional or non-functional) can be common to all products in the product line (this is then implemented as part of the platform). Variability is when a characteristic may be common to some products, but no to all. It must then be explicitly modeled as a possible variability and must be implemented in a way that allows having it in selected products only. Product-specific is when a characteristic may be part of only one

product (at least for the foreseeable future); while these variabilities will not be integrated into the platform, the platform must be able to support them. Thus, variability is a key aspect of SPLs. If the measure takes into account variability, it is classified as *Yes*. Otherwise, it is classified as *No*.

Criterion 8. Validation procedure. It is widely acknowledged that a software measure must be validated according two complementary validations: (1) Theoretical validation, which ensures that the measure measures the attribute that it is supposed to measure, and (2) Empirical validation, which provides evidence on the usefulness of the measures.

The theoretical validation is generally carried out using measurement frameworks based on measurement-theory or other property-based approaches. Property-based approaches [11] allow one to prove that a measure satisfies properties characterizing a concept (e.g., size, complexity, coupling). Measurement-theory based approaches [46] [58] are more rigorous than property-based approaches since they prescribe theories and conditions for modeling and defining measures. The theory provides an empirical interpretation of the numbers (of software measures) by the hypothetical empirical relational system.

The empirical validation of a software measure can be carried out through case studies, surveys, or controlled experiments. A *Case Study* is an observational study and data is collected for a specific purpose throughout the study. A *Survey* is research performed in retrospect, when the method has been in use for a certain period of time. A *Controlled Experiment* is a formal, rigorous, and controlled study. Experiments provide a high level of control and are useful for validating software measures.

Thus, we have classified each measure as follows:

- Theoretical validation: *Yes*, if the measure was theoretically validated, and *No* if the measure was not theoretically validated. If the measure was theoretically validated, we gather the type of approach which was used: *Property-based* approach, *Measurement-theory* based approach, or *Other*.
- Empirical validation: *Yes*, if the measure was empirically validated, and *No* if the measure was not empirically validated. If the measure was empirically validated, we gather the method which was used: *Case Study*, *Survey* or *Controlled Experiment.*
- Not validated: if the measure was not theoretically or empirically validated.

Criterion 9. Tool support. If the measure has a tool that facilitates automatic or semiautomatic measurement, it is classified as *Automatic*. Otherwise, it is classified as *Manual.*

Criterion 10. Current usage. If the measure was proposed in an academic environment and no evidence concerning its actual use is provided, it is classified as *Academic* usage. If the measure was proposed (or is currently being used) in the context of a company, it is classified as *Industrial* usage.

3.2. Conducting the review

The search to identify primary studies in the digital libraries was conducted on the 10th of June, 2010. A total of 305 research papers were obtained from those digital libraries obtained (some papers were duplicated as a result of their appearance in several digital libraries). We also carried out a manual search in the principal software engineering, SPL, and quality conference proceedings. This search did not lead to the discovery of papers which had not already been found with the automatic search. The search of these proceedings did not, therefore, add more papers to the systematic review, but the effectiveness of the search string was confirmed. Finally, we searched grey literature using several combinations of the search string in the Google and Yahoo search engines. This search allowed us to add 6 papers to the total.

The papers were selected in accordance with inclusion and exclusion criteria. The most significant aspect of this selection was that many of the papers from digital libraries were included as potential papers because their titles and abstracts contained the search string. However, once the titles and abstracts had been read (and this sometimes also required scanning the full paper) we discovered

that the topic of the paper was different to that of our study. Finally, 35 papers were selected for our study. Table 5 shows a summary of both potential and selected papers by source.

Table 5. Summary of papers found and selected by source.

Source	# Potential Papers	# Selected Papers (duplicates)	# Selected Papers (not duplicates)
Digital Libraries			
IEEExplore	78	15	15
ACM	73	14	5
Science Direct	36	5	2
Inspec	103	15	7
Springer Link	15	2	0
Grey Literature		6	6
			Total: 35

Since some papers were available in several digital libraries, they appeared in more than one source. Table 5 summarizes the selection of papers by source. It shows that the sources that contributed the greatest number of papers were IEEExplore, ACM Digital Library, and Inspec. The column *Number of Selected Papers (duplicates)* shows the total number of papers selected from each source. Note that the sum of the number of papers in each digital library was greater than the total number of studies that were included in our study. This is because some studies were repeated. Specifically, 18 studies were found in only one source; 12 studies were found in two sources; and 6 studies were found in three sources. The column *Number of Selected Papers (not duplicates)* shows the total number of papers from each source, but in this case we have not considered a paper when it has appeared in a previous source.

4. Results of the systematic review

The systematic review found 35 primary studies, 22 of which presented measures and quality attributes. In these papers we gathered 165 measures that evaluate 98 different quality attributes. The remainder (13 papers) solely discuss quality attributes but do not explain what measures(s) can be used.

The results of the review are discussed in the following subsections. Section 4.1 presents and discusses the results of the data extraction process. Section 4.2 presents statistics concerning the results related to the quality attributes (e.g. number of measures per quality attribute). The appendices show all the results. Appendix A lists the references of the primary studies. Appendix B shows all the measures with each data extraction criteria along with the quality attribute related with these measures. Finally, Appendix C shows the quality attributes (for each quality attribute it shows how many times the quality attribute has been named by an author in the primary studies and the number of measures that measured it).

4.1. Measures for evaluating quality in software product lines

In this subsection we discuss each data extraction criterion and the results obtained. Table 6 summarizes the number of measures and the percentages for each data extraction criterion. Moreover, Appendix B shows all the measures and their classification.

Table 6. Data extraction criteria results.

Data Criteria Strategy	Possible answers	# Measures	Percentage
Criterion 1. Quality characteristic evaluated			
	Functionality Suitability	2	1%
	Reliability	1	1%
	Performance efficiency	9	5%
	Operability	1	1%
	Security	1	1%
	Compatibility	0	0%
	Maintainability	151	92%
	Transferability	0	0%
Criterion 2. Quality attribute evaluated by the measure			
2.1. Type of attribute	Internal attribute	144	87%
	External attribute	21	13%
Criterion 3. Type of measure			
	Base Measure	47	28%
	Derived Meaure	118	72%
Criterion 4. Result of the measure			
4.1. Type of evaluation	Qualitative	2	1%
	Quantitative	163	99%
	Hybrid	0	0%
4.2. Precision	Exact	163	99%
	Probabilistic	2	1%
Criterion 5. Phase of the life cycle in which the measure is applied			
	DE Requirements	16	9%
	DE Design	116	67%
	DE Realization	7	4%
	DE Test	6	3%
	AE Requirements	0	0%
	AE Design	2	1%
	AE Realization	3	2%
	AE Test	4	2%
	Evolution	18	10%
Criterion 6. Artifact(s) used to calculate the measure			
	PL Architecture	110	56%
	Asset	63	32%
	Product architecture	7	4%
	Final Product	17	9%
Criterion 7. Other characteristics			
7.1. Compositionality	Yes	71	43%
	No	94	57%
7.2. Variability	Yes	79	48%
	No	86	52%
Criterion 8. Validation procedure			
Theoretical validation	Yes	13	8%
	No	152	92%
	Property-based approach	13	100%
	Measurement-theory approach	0	0%
	Other	0	0%
Empirical validation	Yes	42	25%
	No	123	75%
	Case study	38	90%
	Survey	0	0%
	Controlled Experiment	4	10%
Not validated		110	67%
Criterion 9. Tool support			
	Manual	80	48%
	Automatic	85	52%
Criterion 10. Current usage			
	Academic	139	84%
	Industrial	26	16%

Following, we discuss each data extraction criterion:

Criterion 1. Quality characteristic evaluated by the measure

The results indicate that most of the measures evaluate attributes related to *maintainability* (92%). In general, these measures are related to variability [6], reusability [61] [53] [48] [56], complexity [61] or evolution [2]. For example, Alves de Oliveira *et al.* [6] present a measure suite to support SPL architecture evaluation. The measures are divided according to the UML element that they measure: use cases, classes, components, diagrams, and UML models representing the overall PL. These measures measure the variability complexity of the UML elements.

Her *et al.* [53] propose a framework for evaluating the reusability of core assets in PL engineering, which consists of five core quality attributes (i.e., Functional commonality, Non-functional commonality, Variability richness, Applicability, Tailorability) and two auxiliary quality attributes (i.e., Understandability, Component replaceability) and their corresponding measures, along with guidelines for applying the measures. Zhang *et al.* [61] measure the complexity of product line architecture by considering variability. Ajila and Dumitrescu [2] propose measures such as Number of modules, Source of change, Code churn, Changes in product line, etc. in order to record changes in the product line and to understand software product line evolution. The rationale for classifying these measures in this quality characteristic is that these describe aspects of the maintainability. SQuaRE [26] classifies reusability as a sub-characteristic of maintainability. We have classified variability as a subcharacteristic of reusability since the reusability in SPL is related with the degree of the variability of the product line. In the literature several studies suggest a relationship between the structural complexity of a software artifact and its maintainability [3]. Moreover, the measures proposed by Zhang *et al.* [61] measure the complexity of product line architecture by considering variability. Finally, the evolution of a SPL [54] is driven by changes in the requirements on the products in the family, and this is partially represented by the modifiability characteristic. In SQuaRE [26], modifiability is a subcharacteristic of maintainability that represents the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Of the total number of measures, 5% measure *efficiency*; such as the Platform efficiency measure and the Efficiency measure for the entire stream of derivate products based on a common product line architecture proposed by Meyer and Dalal [37].

Of the total number of measures, 1% measure quality attributes that are related to the *functional suitability* characteristic. Inoki and Fukazawa [24] proposed Number of kinds of core assets and Total number of core assets measures with which to discover the level of coverage in the SPL.

Of the total number of measures, 1% measure *operability*. Her *et al.* [53] measure the understandibility of core assets by using the Overall Understandability (OU) measure. This measure measures how easily, efficiently, and correctly the core asset description can be comprehended by users, where the core asset description includes the specification, user manual, and any document describing the core asset.

Of the total number of measures, 1% measure *reliability*. Abdelmoez *et al.* [1] measure the probability of an error that arises in one component being propagated to other components, using the Error Propagation measures.

Of the total number of measures, 1% measure *security*. Needham and Jones [40] propose a measure with which to compare the safety represented by the structure and composition of software fault trees with the same root hazard. Unfortunately, we did not find any measure for *compatibility* and *transferability*.

The existing literature has clearly focused on assessing relevant aspects of product lines such as variability and reusability (i.e., quality characteristics included in maintainability). We believe that this fact is produced because these are characteristics very important for SPL. A product that is part of a software product line is created based on the reuse of existing assets. The way the assets are reused depends on the built-in variation mechanisms the assets contain. However, other characteristics are important too in order to assure quality in SPL. There are other measures in literature with which to evaluate quality characteristics, such as those described in the standard ISO 9126 [27], but they are designed to assess individual products and not product lines. It would thus be necessary (1) to theoretically and/or empirically demonstrate that it is possible to use these measures to evaluate

software product lines, or (2) to adapt the existing measures or propose new ones to evaluate other characteristics which are different from maintainability.

Criterion 2. Quality attribute evaluated by the measure

With regard to the **type of attribute**, most of the measures evaluate *internal attributes* (87%), such as the variability complexity [6], the tailorability [53], the evolution [2], the structural soundness [56], which are all related to the product line architecture, structure and its components. For example, the Total number of core assets and the Number of kinds of core assets are examples of measures for the Levels of Coverage [24] attribute.

The *external attributes* are evaluated in 13% of the cases (e.g. the Maturity of an asset is measured in [48] with several measures such as Number of open faults, Number of closed faults, Average of number of days a fault remains, or Average age of a fault). We believe that a final result of a large number of measures that measure internal attributes is a good result, since these attributes can be measured at design time. Moreover, it would be interesting to relate internal attributes to external attributes, since internal attributes may make an impact on external attributes. However, one of the problems that we identified is that we could not find any studies that related internal quality attributes with external quality attributes. Evaluating quality in an efficient manner throughout the whole life cycle of software product lines is a major challenge, and the existence of measures to measure internal attributes would greatly help to achieve this goal.

Criterion 3. Type of measure

The results show that most of the measures are *derived measures* (72%). Only 28% of the measures can be calculated without using other measures *(base measures)*. This is because, in general, the attributes are complex and cannot be calculated directly. Base measures are often used to count the number of elements (e.g., Number of class alternative inclusive variants, Number of use case alternative exclusive variants, Number of variabilities in a class diagram, etc. [1]).

Some measures presented here are the basis for calculating other derived measures. For example, Alves de Oliveira *et al.* [1] define the CompVariant measure as a complexity measure (WMC value) for a given class, and the CompVariantVP as the CompVariant value of the class that is a variation point. They then defined CompVP as CompVariantVP plus the summation of the CompVariant of every Variant (CompVP = CompVariant + (summation from 1 to nVariants (CompVariant))). In [56], Van der Hoek *et al.* present the Pactual and Ptotal measures. Pactual is the number of services provided by a component x that is actually used by other components in the architecture and Ptotal is the total number of services provided by the component x. The authors use these measures to propose the Provided Service Utilization measure (PSU) for individual components ($PSU_{(x)}$ is Pactual divided into Ptotal) and Compound PSU (CPSU) for architectures as a whole (CPSU is defined as the sum of Pactual of all components in the architecture, divided by the sum of Ptotal of all components in the architecture).

Criterion 4. Result of the measure

With regard to the **type of evaluation**, in general, the measures are calculated in a *quantitative manner* (99%). For example, Platform efficiency [37], Relative cost to test a non-generic component [22], and Tracking degradation [28] are calculated by assigning numbers. The measures are calculated in a *qualitative manner* in 1% of cases. For example, [56] use the Compound Provided Service Utilization and Compound Required Service Utilization measures to evaluate the internal cohesion of an architecture by assigning the following values: unbalanced architecture; architecture significantly degraded functionality; the architecture is too large, etc. Finally, none of the measures found provided hybrid results. We believe that having most of the measures with quantitative results is more comfortable for the evaluators as they can establish the values accepted for specific SPL (and for a specific domain).

With regard to the **precision**, most of the measures provided *exact measurement results* (99%). Only 1% of the measures provided *probabilistic results*. For example, Shaik [52] and Abdelmoez *et al*. [1] calculate the Change propagation probability and the Error Propagation Probability of product line architectures, respectively. While having many measures with exact results is beneficial, we think that

it would be advantageous to have probabilistic measures which had different results as a consequence of several parameters.

Criterion 5. Phase of the life cycle in which the measure is obtained

Most of the measures are obtained during the *Design stage of the Domain Engineering phase* (67%); for example, the Functional Coverage, the Coverage of Variability, the Cumulative Applicability and the Component Compliance measures proposed by SunHer [53]. Of the total number of measures, 9% were related to the *Requirements stage of the Domain Engineering phase*. Some examples of measures that can be applied in this phase include the measures proposed by Alves de Oliveira [1]: number of variation points in a use case diagram; number of alternative inclusive variants in a use case diagram; number of alternative exclusive variants in a use case diagram; number of optional variants in a use case diagram; number of variabilities in a use case diagram. This is a good result from the point of view of the design. Of the total measures, 4% were related to the *Realization stage of the Domain Engineering phase*. Some measures that can be applied in this phase are: Clone Coverage [60], Number of Readable Properties, Number of Writable Properties [48]). Another 3% of the total is obtained during the *Test stage of the Domain Engineering phase*. Some measures that can be applied in this phase are the Number of Open Faults, Average of Number of Days to Close a Fault, and Average of Number of Age of a Fault proposed in [48]).

Although we found a great number of measures, few of them can be applied in the *Application Domain Phase*. Specifically, there is no measure that can be applied in the *Requirements of the Application Domain Phase*; only 1% of them can be applied in the *Design phase* (e.g. Software Fault Tree measure [40], Component Reuse Rate [61]); only 2% of them can be applied in the *Realization phase* (e.g., Maintainability Index [2]), and only 2% of them can be applied in the *Test phase* (e.g. Tracking Degradation [28]).

It is widely accepted that a good design improves the final software and decreases future errors. Maybe for this reason most of the measures are focused on the Design stage of the Domain Engineering phase. However, the quality assessment in the following phases of the SPL life cycle is equally important since the quality in these phases could be modified when new requirements are introduced. We believe that it would be desirable to have measures with which to validate a SPL in all the phases of its construction, rather than being centered on only a few stages of its life cycle. The results therefore show that more measures are needed to cover more phases of the SPL life cycle. Finally, 10% of the measures measure the SPL in the *Evolution Phase* (e.g. Impact of change, Adjusted product line growth, and Source of change measures proposed by [2]).

Criterion 6. Artifact(s) used to calculate the measure

The results show that the first most frequently evaluated artifact is *Common architecture* (56%) (e.g., [61], who propose PLA-IFG Cyclomatic Complexity, PLA-IFG vertex complexity, or PLA-IFG total complexity, and [6], who propose Number of variabilities in use cases of a PL, Number of variabilities in classes of a PL, or Number of variabilities in a PL). This result is consistent with the phase of the life cycle that is most frequently evaluated: the Design in the Domain Engineering phase. Furthermore, we believe that this is a good result for the challenges posed by product lines. By ensuring the quality of the common architecture this quality can be inherited by the product architecture. The second most frequently evaluated artifact is the *asset* (32%). For example, the measures proposed by Rahman [48] for Interface Complexity (Number of properties, Number of services, Number of events, Number of pre-conditions, Number of postconditions, Number of distinct range constraints on properties, etc.). The quality of the assets is also very important since the final product will be built by using a set of assets. Of the total, 9% are related to the *Final Product* artifact (e.g. Binary Size, Performance or Cyclomatic complexity [50]). The *product architecture* artifact is evaluated by 4% of the measures (e.g., the Component Reuse Rate [61], or Software Fault Tree Metric [53]).

Criterion 7. Other characteristics

The results show that 43% of the measures take into account issues related to the *compositionality*. For example, the measures proposed by Ganesan *et. al*. [22] evaluate issues related to the component, types of components and the relationship between them. On the other hand, there are other measures

such as Binary Size, Performance and Cyclomatic Complexity proposed in [50] that *do not take the compositionality* into consideration.

The situation with regards to the *variability* is very similar: 48% of the measures take into account issues related to the variability whereas 52% do not. For example, the variability is considered in the measures proposed by Sun Her [53], Alves de Oliveira [6], and Zhang [61]. But others like Code churn, Size of code in the product line or Impact of change [2]; Efficiency for any single derivate product or Reuse for any single product [37]; or Change propagation probability [52] do not.

We believe that, depending on the quality attribute (e.g., maintainability, reusability, modifiability) and the life cycle phase (e.g., requirements in domain engineering, design in domain engineering) in which the measure is evaluated, the compositionality and the variability properties may be more or less important. For instance, variability may be more important in the reusability for an asset in the domain engineering design phase than for a final product in the application engineering testing life cycle. Thus, we believe that when the measures are defined, they should consider these properties.

Criterion 8. Validation procedure

The results show than 67% of the measures are not validated. With regard the validated measures, 13 measures (8%) have been theoretically validated and 42 measures (25%) have been empirically validated.

With regard the measures validated theoretically, all of them use a Property-based approach. For instance Sun Her *et al.* [53] included a limited theoretical analysis based on the framework proposed by Kitchenham [33] as well as an analysis of conformance using some criteria described by Ejiogu [19] and IEEE Std 1061-1998 [18].

With regard the measures validated empirically, 90% of them use Case Studies and 10% use Controlled Experiments. For instance, Wnuk *et al.* [59] evaluate the four measures (e.g., number of scope inclusions at the timestamp, number of days needed to make a final decision about feature extension) in three large platform projects and one measure (Reasons for scope exclusions) in one large project. Sethi *et al.* [49] compare the conclusions reached by their measures with the conclusions obtained from previous researchers for the same system.

The validation confirms that the measure is measuring that it intends to measure and the results are as expected. Thus, although the systematic review has found many measures, there is a need to validate them in order to verify them. Moreover, we want to comment that the significance and relevance of the methods for software validation is often confusing. For example, many authors claim to perform a case study when in fact they are making a proof of concept of their proposal. We intend to differentiate between the two cases. Thus, we classified the measures as *Not validated*, as their validations were carried out using an incorrect design of the validation. Unfortunately, most of the papers do not fully explain the validation process and results. As such, it is difficult to know if a good design exists or if it is only a proof of concept or a short validation.

Criterion 9. Support for the measure

The results show that many of the measures *do not have a tool* to support their evaluation (48%). In some cases, the tool offers a semi-automatic evaluation. In these cases, we have classified these measures as being automatic because there is a tool to assist the user. In total, 52% of the measures have a tool which assists in the evaluation. Among the tools available are SD Metrics and the Eclipse plug-in [6], SemanticDesigns [2], Matlab [2], Architecture Change Propagation Tool (SACPT) [52].

Criterion 10. Current usage

The results show that the authors of the articles commonly belong to universities. The use of the measure is therefore often *academic* (84%). However, sometimes the measure has been adapted for *industrial* use (16%) (e.g., Testo AG [22] or Ericcson Mobile Communication AB [28]). Perhaps future research should attempt to determine why quality assurance measures are not being widely used by the industrial sector. Nevertheless, our study is focused on papers and journals published in research forums. This may be a limitation of this study because there might be other measures being used in industrial settings.

4.2. Quality Attributes for Software Product Lines

Quality attributes form the basis for product line quality evaluation. However, they should be precisely defined because without elaboration quality attributes are subject to interpretation and misunderstanding. Quality attributes exist in the context of specific goals (e.g., a system is modifiable or not with respect to a specific type of change). The first step in defining a measure must be to establish which quality attribute it is suppose to measure (and why it is relevant to measure it). In section 4.1, we have summarized the software measures that have been proposed by researchers and we have related them with their measured quality attribute. In some cases the authors of the papers explicitly state the quality attribute. But in others, the authors do not relate the quality attributes with the proposed measure(s) (the measures are proposed but no explanation about the quality attributes are provided). We want to highlight this issue because it is important to specify the quality attribute measured when a measure is proposed. In these cases, we have attempted to infer the quality attribute from the context.

Moreover, some authors have discussed the importance of quality attributes for SPLs. We have found 13 papers that discuss 76 quality attributes but do not propose measures for measuring them. This is a good reference through which to discover which attributes are considered to be important by other authors. Also, this information can be useful to provide quality attributes for the measures presented in Section 4.1 (only few studies indicated the quality attribute being measured). Although the authors of these papers do not present a measure in their works, some of these attributes have had a measure proposed in other papers which allows them to be measured. We do not know why these authors do not use other existing measures. Software engineers often need to express the artifact in a specific model or language to use a measure. Perhaps these authors cannot use existing measures in their works.

The table shown in Appendix C summarizes all the attributes founded in this systematic review. Each quality attribute may appear in several papers. This quantity is shown in the Number of repetitions column. In addition, each quality attribute may have several measures with which to measure it (or no measure). This quantity is shown in the Number of measures column.

We have attempted to classify the attributes without measures in several criteria (characteristic of quality, type of attribute, phase of life cycle), but the majority of authors do not correctly explain all the information. The attributes with measures are classified in Section 4.1 (Criterion 2) together with their measures. Moreover, the table shown in Appendix B shows the quality attribute measured by each measure.

The results show that there are 98 quality attributes that have measures (165 measures in total) and 76 quality attributes that do not have measures. There are thus many quality attributes that do not have measures. The quality attributes with the greatest number of measures are: Complexity of a class (12 measures), Complexity of a class diagram (9 measures), Complexity of a use case (9 measures), Effort (7 measures), Complexity of a use case diagram (6 measures), Maturity (6 measures), or Structural Soundness (6 measures).

From our analysis we detected that there are quality attributes that are relevant for any type of product lines since they have an impact on the product line architectural level (e.g., modifiability, variability). However, there are several other quality attributes that are very specific from the domain. Considering the information extracted from the papers, it is not clear what the relevance is of these quality attributes in practice. In general, the information provided in the papers about the quality attributes is very poor (e.g., no scenario to illustrate the quality attributes was found). As future work we plan to validate the relative importance of these quality attributes for specific domains (e.g., embedded systems, safety-critical systems) to provide more guidance to practitioners.

5. Discussion

In this section, we discuss the relevance and the contribution of the results of the systematic review, and we point out the strengths and weaknesses of the evidence gathered. Since we wish to ensure the validity of our systematic review, we also discuss its possible limitations and how they can be

resolved. Furthermore, we explain the validation of the review protocol using the procedure described in Section 3.1.1.

5.1. Strengths and weaknesses

The main strengths are related to the life cycle development phase and the type of attribute evaluated. With regard to the life cycle development phase, many measures are used in early stages of the life cycle for evaluating specific properties of the product line architecture. However, regardless of this result, we still believe that new measures should be proposed to cover the entire life cycle. With regard to the attribute evaluated, many measures evaluate attributes that are relevant for SPL. We believe that this is a good result since SPL is a development approach that is quite new, and there are fewer methods and techniques for its evaluation than those available for the development of individual products.

The main weaknesses are related to: the difficult application and adaptation of the measures to any given SPL; the absence of measures that evaluate all of the quality characteristics; and the need to validate the measures. With regard to the difficult application and adaptation of the measures, we have found a large number of measures; however, it is not generally possible to select one and implement it quickly. The main problem is that the measures are not correctly specified and/or do not specify the attribute evaluated. For example, Alves de Oliveira *et al*. [6] propose a large number of measures. They indicate 35 measures to support the evaluation of the product line architecture and also indicate that combinations of these measures that can be used to calculate attributes such as complexity, maintainability, and testability. However, the process is not clear. Some of the attributes measured are not defined (they proposed 35 measures but they do not indicate the quality attribute measured by each measure), and some of the measures are not clearly defined. For example, for the measure UseCaseOptional [6] only a brief description is provided ("Use case is an optional variant"). The authors do not clearly define what quality attribute this measure is supposed to measure neither how it can be measured. Another aspect to consider is the type of specification or notation used to represent the artifacts. Depending on the specification used, some measures can be used and others cannot. For example, the Software Fault Tree measure [40] uses a representation with logic gates, the measures proposed by [50] or [2] use the source code, and the measures proposed by [61] to specify the product line architecture with using vADL language (a product line architecture description language) that was designed for this purpose. Each measure is thus defined for a particular specification of the artifact and it is not possible to use all of measures definitions in different artifacts.

With regard to the absence of measures that evaluate all quality characteristics, we have proved in the results that most of the measures evaluate attributes that are related to the maintainability characteristic. This characteristic is relevant for SPL because it is related to subcharacteristics such as reusability, compositionality, and complexity. However, the remaining quality characteristics are also important for SPL engineering.

With regard to the need to validate the measures, quality is an important issue since these measures are intended to assess the quality of software product lines; we believe that they should also be evaluated. The validation of a measure ensures that the measure evaluates the attribute correctly. This undertaking gives the measures more relevance. However, according to the systematic review only 31% of the measures have been validated. However, many authors confuse case study validation with proof of concept, which are different actions. Moreover, none of the measures from our review were tested in a controlled experiment to study the effects of internal quality attributes on external quality attributes.

Among the measures reviewed, we highlight those of [53], who propose a framework for evaluating the reusability of core assets in product lines. This study presents a clear statement of the proposed measures, the artifacts used, and the relations among them. Moreover, the authors attempt to perform a theoretical validation of the proposed framework.

5.2. Implications for research and practice

The contributions of this paper have implications for both research and practice. The review shows the current state of the art with a certain level of guaranteed quality in the results. Knowing the state of the art helps us to detect deficiencies, direct efforts, and identify new lines of research.

For both researchers and practitioners, the result of this systematic review is a catalogue of measures which will enable them to discover which measures might be useful to them, which phase of the life cycle the measure should be applied to, and which artifact can be evaluated. Our study also shows whether the measure can be directly applied or whether it must be adapted to the SPL. The need for new quality attributes or measures can also be detected.

The measures and quality attributes collected in this study are a good starting point for defining a quality model for SPL. As defined in ISO/IEC 25000 (SQuaRE) [26], a quality model is a set of characteristics and their relationships with each other, which form a framework for specifying requirements and evaluating quality. Both researchers and practitioners can save time and effort since they can reuse the measures or adapt them to other artifacts or specifications. Moreover, new measures could be defined to measure the quality attributes that have been identified in this study and that do not have any measure; or to cover the gaps that have been identified in the results of the systematic review. There is a lack of quality attributes and measures related to functional suitability, reliability, performance efficiency, operability, security, compatibility, and transferability. Additionally, it is important to validate both existing measures and new proposed measures.

It would also be of interest to define a catalog of measures for specific domains with information about each of them (e.g., quality attribute evaluated, the definition of the measure, the artifact evaluated, and how to apply the measure with these artifacts) and classify them taking into account the relative importance for the domain.

6. Validation of the systematic review

The main limitations of this study concern publication selection bias, incomplete string search, and inaccuracy in data extraction or misclassification. To ensure that the systematic review was as correct, complete and objective as possible, in this section, we discuss the threats to validity and the actions taken to validate the planning of the systematic review protocol, the selection of primary studies, and the data extraction strategy.

Before validating the planning, we reviewed the possible limitations. We thus selected digital libraries that contain a very large set of publications in the Software Engineering field (IEEExplore, ACM Digital Library, Science Direct, Springer Link and Inspec). Moreover, we scanned relevant conference proceedings and journals. We chose those sources in which studies concerning software quality, SPL and software engineering are normally published.

With regard to the search string, we attempted to collect all the strings that were representative of the two research questions used. We refined the search string on several occasions based on the results obtained in order to maximize the selection of papers related to the systematic review. We additionally considered synonyms and included the lexeme of words. We also applied term patterns and adapted the search string to each digital library in order to make the replication of the process easier. Then, we ensured that the studies with which we were familiar were in the results. In addition, we included grey literature. However, we did not consider unpublished results or papers not written in English.

6.1. Validation of the review protocol

In order to evaluate the systematic review protocol we analyzed several guidelines to ensure that we had included all the important activities. Once the guidelines to evaluate the review activities had been established, we formulated the questions shown in Table 4. These were given to five domain experts in the field of software engineering, who reviewed the planning and answered the questions proposed for evaluating the protocol.

Table 4. Questionnaire for validating the planning of the systematic review

Issues concerning the final search string:	5 4 3 2 1
▪ The search string has sufficient synonyms.	☐☐☐☐☐
▪ The search string has sufficient lexemes.	☐☐☐☐☐
▪ The search string is too generic. (It obtains too many studies)	☐☐☐☐☐
▪ Familiar studies that are representative of the quality in SPL appear in the results of the automatic search. (Example list of SPL papers) - Van der Hoek et al. "Using Service Utilization Metrics to Assess the Structure of Product Line Architectures". IEEExplore. - D. Needham and S. Jones. "A Software Fault Tree Metric". IEEExplore. - T. Zhang et al. "Some Metrics for Accessing Quality of Product Line Architecture". IEEExplore. - N. Siegmund. "Measuring Non-functional Properties in Software Product Lines for Product Derivation". IEEExplore. - J. S. Her. "A framework for evaluating reusability of core asset in product line engineering". ACM.	☐☐☐☐☐
Issues concerning the primary sources:	**5 4 3 2 1**
▪ The digital libraries are representative of the area of study.	☐☐☐☐☐
▪ The conference and journals selected for the manual search are representative of the area of study.	☐☐☐☐☐
Issues concerning the extraction data criteria:	**5 4 3 2 1**
▪ The two research questions are fully answered using the established criteria.	☐☐☐☐☐
▪ There are other criteria that could be of interest. Indicate which ones:	☐☐☐☐☐
▪ The descriptions of the criteria are clear.	☐☐☐☐☐

Each question has a descending scale of five possible values. The value of 5 signifies complete agreement with the statement. The value of 1 signifies complete disagreement with the statement. The empirical study consists of calculating the average of all the domain experts. If the final result is close to 5, we consider that the planning protocol has an appropriate level of quality. If the result is close to 1, it must be reviewed.

The validation obtained a mark of 4.8 points of a possible 5. Although we believe that this is a good result, we have considered the feedback of these experts in order to make improvements. The main improvements were to introduce QoSA and RefsQ conference into the manual search, to change the name of several data extraction criteria in order to improve their understanding, and to establish a more accurate definition for certain data extraction criteria

6.2. Validation of selection of primary studies

In order to validate the correct descriptions of the inclusion and exclusion criteria, and the correct selection of the studies by the reviewers, we did the following:

1. We selected a random set of ten studies (from which 6 studies were included and 4 studies were excluded).
2. The three reviewers of this research selected the studies by using the inclusion and exclusion criteria.
3. The results were analyzed with Fleiss' kappa.

Fleiss' kappa is a statistical measure for assessing the reliability of agreement between a fixed numbers of raters when assigning categorical ratings to a number of items or classifying items. The measure calculates the degree of agreement in classification over that which would be expected by chance and is scored as a number between 0 and 1 (0 is a low agreement and 1 is a high agreement). We used the Fleiss' kappa to validate the process of inclusion/exclusion of primary studies. Finally,

the results of the selection of each reviewer were checked and the discrepancies were resolved with consensus.

Fleis's kappa for agreement on inclusion in the review was 0.86 for the three raters. Although there is not yet a general agreement on measure of significance, Landis and Koch [34] provide a table for interpreting the values. In this table, they interpret that values between 0.81 and 1.00 are considered Almost perfect agreement. Overall, this suggests excellent agreement among our raters.

6.3. Quality assessment of the primary studies

We have checked the relevance of the primary studies to provide their quality assessment. Each study is rated according to two issues: (1) relevance of the conference or journal where the paper was published, and (2) number of citations of the paper.

With regard the relevance of the conference or journal where the paper was published, we have classified the papers in three categories: "Very relevant", "Relevant", and "Not so relevant". This question was rated by considering the CORE conference ranking (A, B, and C conferences) [15], and the Journal Citations Reports (JCR) lists [29]. The following shows how the papers were classified:

- **Very relevant**: papers published in conferences rated as A in the CORE classification or published in journals included in the JCR lists. In addition, we included in this category the papers from conferences dedicates to SPL (e.g., Software Product Line Conference, Software Product Families Engineering) since these conferences do not appear in the CORE classification but they are very relevant for the SPL area. They were scored with 10 points.
- **Relevant**: papers published in conferences ratted as B or C in the CORE classification or published in journals not included in the JCR lists. In addition, we included in this category thesis and technical reports. They were scored with 5 points.
- **Not so relevant**: papers published in conferences not indexed in the CORE classification. They were scored with 0 points.

With regard to the number of citations of the paper, we classified again the papers in three categories: "High", "Medium", and "Low". This question was rated by considering the Google scholar citation count. Note that we classified in a different way the papers published before 2008 and the rest of papers. The reason is to not penalize the early publications. The following shows how the papers were classified:

- If the paper was published before 2008:
  - **High**: papers with more than 5 citations. They were scored with 10 points
  - **Medium**: papers cited by 1-5 authors. They were scored with 5 points.
  - **Low**: papers have not been cited. They were scored with 0 points.

- If the paper was published in or after 2008:
  - **Potentially High**: papers cited. They were scored with 10 points
  - **Potentially Medium (n/d)**: papers have not been cited. They were scored with 5 points.

For each paper we have calculated these two aspects. These scores were not used to exclude papers in the systematic review due the fact our objective is to gathering all the quality attributes considered relevant for SPL and the measures proposed for SPL. However, these results address the quality of the primary studies of our study.

The average of the first quality assessment (relevance of the conference or journal where the paper is published) is 7.14 points. Whilst, the average of the second quality assessment (number of citations of the paper) is 8.14 points. The results show that the selected papers are, generally,

published in relevant conferences and/or journals and they have been referenced by several other publications.

6.4. Validation of data extraction criteria and classification

The threats to validity with regard to data extraction criteria and classification are related to an unclear description of the data extraction criteria and problems with misclassification. With regard to the description of the data extraction criteria, it is important to ensure that the definition is clear. A confusing description may cause problems in the classification.

The data extraction criteria were validated by following similar steps to those described in Section 6.2. First, we selected a random set of 3 studies (corresponding with 48 measures). Then, three reviewers classified the studies with the data extraction criteria, and the results were analyzed with Fleiss' kappa. The results of the selection of each reviewer were checked and the discrepancies were resolved by consensus.

Fleis's kappa for agreement in the review was 0.79. Landis and Koch [34] interpret values between 0.61 to 0.80 as *Substantial agreement*. Note that the value is very close to the next category: *Almost perfect agreement*. The classification was not easy since the studies did not provide clear answers to our data extraction criteria. Many measures were not described correctly or others were not related to a quality attribute. Upon considering these drawbacks, we believe that this result provides a good coefficient of agreement.

7. Conclusions and future work

Quality is a crucial factor in software engineering, but it is more important in SPL because an error in the common architecture or in core assets may be propagated to final products. When evaluating quality, measures are a good approach through which to predict, control and evaluate the software. A great number of measures for evaluating SPL exist in literature. However, no study that gathers all the relevant information about this topic exists.

We have presented a systematic literature review with the aim of locating and obtaining relevant information about the measures and quality attributes used by researchers from 1996 to 2010 to assess the quality of SPL. We have identified 165 measures and 174 quality attributes. Of 174 quality attributes, 98 are evaluated for one or more measures, that is to say, 76 quality attributes found do not have any measure for measuring them.

With regard to the measures, the most significant result is that 67% are not validated. We want to point out that many authors claim that their measures are empirically validated through a case study, but in fact they are using the case study as a proof of concept to show how the measures works. Validation is an essential process to ensure the usefulness of software measures and it is important to emphasize the relevance of the validation methods. A further significant result is that a high number of measures were proposed for evaluating the maintainability characteristic (92%), whilst the remaining quality characteristics are drastically ignored.

With regard to the life cycle phase in which the measures are applied, the study shows that the measures focus on measuring artifacts produced during the requirements of domain engineering (9%), the design of domain engineering (67%) and evolution (10%). That is to say, most of the measures focus on the early phases of life cycle (requirements and design of domain engineering). In SPL, it is important to evaluate the quality in these phases, since the common architecture and core assets are designed in these phases. However, new requirements and assets, unpredictable restrictions between assets etc. may appear. In summary, errors may be introduced in other phases, so it is important to ensure quality in the whole life cycle.

The results obtained show that there is a need to propose more measures in order to cover different gaps (e.g., to measure more quality attributes, quality characteristics, in more phases of the life cycle), and the measures must be correctly validated.

We hope that our study will be used by both researchers and practitioners. The study is of interest to researchers since they may find gaps in this topic and suggest issues to be further investigated, whereas practitioners will be able to discover the current state of the art and recognize measures which can be used/adapted in their companies. Only 16% of the measure findings are used in

an industrial environment. We believe that it is important to transfer and share knowledge between companies and academic researchers.

As future work, we are defining a quality model for SPLs from the decomposition of the quality model proposed in the standard SQuaRE [26]. This will integrate the measures obtained in this work and add others necessary to complete the model. We have recently presented a preliminary version of our quality model in [39]. Since most of the measures founded in this systematic review are not validated, we will also conduct empirical studies to validate the usefulness of selected measures.

Acknowledgements. This research has been funded by the Spanish Ministry of Science and Technology under the MULTIPLE (Multimodeling Approach For Quality-Aware Software Product Lines) project with ref. TIN2009-13838.

References

[1] Abdelmoez, W., Nassar, D.M., Shereschevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B., Mili, A.: Error Propagation In Software Architectures. In 10th International Symposium on Software Metrics (METRICS), Chicago, Illinois, USA, 2004.

[2] Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. Journal of Systems and Software 80, pp.74-91, 2007.

[3] Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O. Experience measuring maintainability in software product lines. In XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD). Barcelona, 2006.

[4] Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O.: Quantifying Maintanibility in Feature Oriented Product Lines, Athens, Greece, pp.243-247, 2008.

[5] Alves, V., Niu, N., Alves, C., Valença, G. Requirements engineering for software product lines: A systematic literature review. Information & Software Technology 52(8): 806-820 (2010)

[6] Alves de Oliveira Junior, E., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In XXXIV Conferencia Latinamericana de Informática (CLEI), Santa Fé, Argentina, pp.489-498, 2008.

[7] Bachmann, F. & Clements, P. Variability in Software Product Lines (CMU/SEI-2005-TR-012, ADA450337). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[8] Benavides, D., Segura, S., Trinidad, P., and Ruiz-Cortés, A. "FAMA: Tooling a Framework for the automated analysis of feature models". In 1st International Workshop on Variability Modelling of Software Intensive Systems, 2007, pp.129-134.

[9] Bosch, J.: Design and use of software architectures: adopting and evolving a product line approach. ACM Press/Addison-Wesley Publishing Co., USA, 2000.

[10] Briand, L.C., Differing, C.M., Rombach, D. Practical Guidelines for Measurement-Based Process Improvement. Software Process-Improvement and Practice 2, pp.253-280, 1996.

[11] Briand, L.C., Morasca, S., and Basili, V.R. Property Based Software Engineering Measurement. IEEE Trans. Software Eng., vol. 22, no. 1, pp. 68-86, Jan. 1996.

[12] Calero, C., Ruiz, J., Piattini, M.: Classifying web metrics using the Web Quality Model. Online Information Review, OIR - 29, 3 Emerald Literari. United Kingdom.

[13] Chen, L., Ali Babar, M., Ali, N. Variability Management in Software Product Lines: A Systematic Review. In 13th International Software Product Lines Conferences (SPLC), San Francisco, USA, 2009.

[14] Clements, P., and Northrop, L. Software Product Lines. 2003. Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, MA, 2002.

[15] Conference Rankings of Computing Research and Education Association of Australasia (CORE). Available in http://core.edu.au/index.php/categories/conference%20rankings/1

[16] Crnkovic, I., Larsson, M. Classification of Quality Attributes for Predictability in Component-Based Systems. Journal of Econometrics, pp.231-250, 2004.

[17] Davis, A., Dieste, Ó., Hickey, A., Juristo, N., Moreno, A.M. Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. In 14th IEEE International Conference Requirements Engineering, pp.179-188, 2006.

[18] IEEE standard for a software quality metrics methodology, IEEE Std 1061-1998, 1998.

[19] Ejiogu, L. Software Engineering with Formal Metrics. QED Publishing, 1991.

[20] Engström, E., Runeson, P. Software product line testing - A systematic mapping study. Information & Software Technology 53(1): 2-13 (2011).

[21] Etxeberria, L., Sagarui, G., Belategi, L. Quality aware software product line engineering. Journal of the Brazilian Computer Society, vol.14, no.1, Campinas Mar, 2008.

[22] Ganesan, D., Knodel, J., Kolb, R., Haury, U., Meier, G.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. In 11th International Software Product Line Conference, Kyoto, Japan, pp.74-83, September 2007.

[23] Gómez, O., Oktaba, H., Piattini, M., García, F.: A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures. In First International Conference on Software and Data Technologies (ICSOFT), Setúbal, Portugal, pp.11-14. September, 2006.

[24] Inoki, M., Fukazawa, Y.: Software Product Line Evolution Method Based on Kaizen Approach. In 22nd Annual ACM Symposium on Applied Computing, Korea, 2007.

[25] Insfran, E., Fernandez, A. A systematic review of usability evaluation in Web development. 2nd International Workshop on Web Usability and Accessibility (IWWUA'08), New Zeland, LNCS 5176, Springer, pp.81-91, 2008.

[26] ISO/IEC 25000:2005. Software Engineering. Software product Quality Requirements and Evaluation (SQuaRE).

[27] ISO/IEC 9126. Software Engineering. Product Quality.

[28] Johansson, E., Höst, R.: Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations. In 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, pp.249-254, 2002.

[29] Journal Citation Reports of Thomson Reuters. Available in http://thomsonreuters.com/products_services/science/science_products/a-z/journal_citation_reports/

[30] Kim, T., Ko, I.Y., Kang, S.W., and Lee, D.H. “Extending ATAM to assess product line architecture”. In 8th IEEE International Conference on Computer and Information Technology, pp. 790-797, 2008.

[31] Khurum, M., Gorschek, T. A systematic review of domain analysis solutions for product lines. The Journal of Systems and Software. 2009.

[32] Kitchenham, B. “Guidelines for Performing Systematic Literature Reviews in Software Engineering”. Version 2.3, EBSE Technical Report, Keele University, UK.

[33] Kitchenham, B., Pfleeger, S., Fenton, N.: Towards a framework for software measurement validation. IEEE Transactions on Software Engineering 21 (12), 1995.

[34] Landis, J. R. and Koch, G. G. (1977) "The measurement of observer agreement for categorical data" in Biometrics. Vol. 33, pp. 159–174.

[35] Matinlassi, M., Niemelä, E., and Dobrica, L. “Quality-driven architecture design and quality analysis method: A revolutionary initiation approach to a product line architecture”. Technical Report VTT-PUBS-456, VTT, 2002.

[36] Mendes, E.: A systematic review of Web engineering research. International Symposium on Empirical Software Engineering. 2005. Noosa Heads, Australia.

[37] Meyer, M. H., and Dalal, D. Managing platform architectures and manufacturing processes for non assembled products. Journal of Product Innovation Management. Volume 19, Issue 4, pages 277–293, July 2002.

[38] Montagud, S., Abrahão, S. Gathering Current Knowledge about Quality Evaluation in Software Product Lines. In 13th International Software Product Lines Conferences (SPLC), San Francisco, USA, 2009.

[39] Montagud, S. Abrahão, S. A SQuaRE-bassed Quality Evaluation Method for Software Product Lines. Master’s thesis, December 2009 (in Spanish).

[40] Needham, D., Jones, S.: A Software Fault Tree Metric. In 22nd International Conference on Software Maintenance (ICSM), Philadelphia, Pennsylvania, USA, 2006.

[41] Niemelä E., and Immonen, A. “Capturing quality requirements of product family architecture”. Information and Software Technology, 2007, vol. 49(11-12), pp.1107-1120.

[42] Odia, O. E. Testing in Software Product Lines. Master Thesis Software Engineering of School of Engineering, Bleking Institute of Technology. Thesis no. MSE-2007:16, Sweeden, 2007.

[43] Olumofin, F. G., and Mišić, V. B. “A holistic architecture assessment method for software product lines”. Information and Software Technology 49, 2007, pp. 309-323.

[44] Pérez Lamancha, B., Polo Usaola, M., Piattini Velthius, M. Software Product Line Testing - A Systematic Review. ICSOFT (1) 2009: 23-30.

[45] Petticrew, Mark and Helen Roberts. Systematic Reviews in the Social Sciences: A Practical Guide, Blackwell Publishing, 2005, ISBN 1405121106.

[46] Poels, G. and Dedene, G. Distance-based software measurement: necessary and sufficient properties for software measures. Information and Software Technology. 42(I), 35-46. 2000.

[47] Prehofer, C., van Gurp, J., Bosch, J. Compositionality in Software Platforms. In Emerging Methods, Technologies and Process Management in Software Engineering, Wiley, 2008.

[48] Rahman, A.: Metrics for the Structural Assessment of Product Line Architecture. Master Tesis on Software Engineering, Thesis no. MSE-2004:24. School of Engineering, Blekinge Institute of Technology, Sweden, 2004.

[49] Sethi, K. Cai, Y., Wong, S., Garcia, A., Sant'Anna, C. From Retrospect to Prospect: Assessing Modularity and Stability from Software Architecture. Joint Working IEEE/IFIP Conference on Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009.

[50] Siegmund, N., Rosenmüller, M., Kuhlemann, M., Kästner, C., Saake, G.: Measuring Non-functional Properties in Software Product Lines for Product Derivation. In 15th Asia-Pacific Software Engineering Conference, Beijing, China, 2008.

[51] de Souza Filho, E. D., de Oliveira Cavalcanti, R., Neiva, D. F. S., Oliveira, T.H.B., Barachisio Lisboa, L., de Almeida E.S., and de Lemos Meira, S. R. "Evaluating Domain Design Approaches Using Systematic Review". In 2nd European Conference on Software Architecture, Cyprus, 2008, pp.50-65.

[52] Shaik I., Abdelmoez, W,. Gunnalan, R., Shereshevsky , M., Zeid, A., Ammar, H.H., Mili, A., and Fuhrman, C. Change Propagation for Assessing Design Quality of Software Architectures. 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005.

[53] Sun Her, J., Hyeok Kim, J., Hun Oh, S., Yul Rhew, S., Dong Kim, S.: A framework for evaluating reusability of core asset in product line engineering. Information and Software Technology 49, pp. 740-760, 2007.

[54] Svahnberg, M., Bosch, J. Evolution in software product lines. In 3rd International Workshop on Software Architectures for Products Families (IWSAPF-3). Las Palmas de Gran Canaria, 2000.

[55] Thiel, S. "On the definition of a framework for an architecting process supporting product family development". In 4th International Workshop on Software Product-Family Engineering, Springer-Verlag, London, UK, 2002, pp.125–142.

[56] Van der Hoek, A., Dincel, E., Medidović, N.: Using Services Utilization Metrics to Assess the Structure of Product Line Architectures. In 9th International Software Metrics Symposium (METRICS), Sydney, Australia, 2003.

[57] Van der Linden, F., Schmid, K., Rommes, E. Software Product Lines in Action. Springer, 2007.

[58] Whitmire, S. Object Oriented Design Measurement. 1997.

[59] Wnuk, K., Regnell, B., Karlsson, L. What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting. In 17th IEEE International Requirements Engineering Conference, 2009.

[60] Yoshimura, K., Ganesan, D., Muthig, D.: Assessing Merge Potential of Existing Engine Control Systems into a Product Line. In International Workshop on Software Engineering for Automative Systems, Shangai, China, pp.61-67, 2006.

[61] Zhang, T., Deng, L., Wu, J., Zhou, Q., Ma, C.: Some Metrics for Accessing Quality of Product Line Architecture. In International Conference on Computer Science and Software Engineering (CSSE), Wuhan, China, pp.500-503, 2008.

Appendix A

List of papers selected in the systematic review.

S01 Abdelmoez, W., Nassar, D.M., Shereschevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B., Mili, A.: Error Propagation In Software Architectures. In 10th International Symposium on Software Metrics (METRICS), Chicago, Illinois, USA, 2004.

S02 Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. Journal of Systems and Software 80, pp.74-91, 2007.

S03 Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O.: Quantifying Maintanibility in Feature Oriented Product Lines. In 12th European Conference on Software Maintenance and Reengineering, Athens, Greece, pp.243-247, 2008.

S04 Alves de Oliveira Junior, E., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In XXXIV Conferencia Latinamericana de Informática (CLEI), Santa Fé, Argentina, pp.489-498, 2008.

S05 Axelsson, J. Evolutionary Architecting of Embedded Automotive Product Lines: An Industrial Case Study. In Joint Working IEEE/IFIP Conference on Software Architecture 2009 and European Conference on Software Architecture 2009, WICSA/ECSA 2009, Cambridge, UK, 14-17 September 2009.

S06 Dincel, E., Medvidovic, N., Van der Hoek, A.: Measuring Product Line Architectures. In 4th International Workshop on Product Family Engineering (PFE), Bilbao, Spain, 2001.

S07 Etxeberria, L., and Sagardui, G. Product-Line Architecture: New Issues for Evaluation. Software Product Lines Conference, 2005,

S08 Gallina, B. and Guelfi, N. A Product Line Perspective for Quality Reuse of Development Framework for Distributed Transactional Applications. In Internacional Computer Software and Applications Conference, 2008

S09 Gallina, B., and Guelfi, N. A Template for Requirement Elicitation of Dependable Product Lines. In International Workshop on Requirements Engineering: Foundation for Software Quality (RefSQ), 2007.

S10 Ganesan, D., Knodel, J., Kolb, R., Haury, U., Meier, G.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. In 11th International Software Product Line Conference, Kyoto, Japan, pp.74-83, September 2007.

S11 Gannod, G.C., and Lutz, R.R. An Approach to Architectural Analysis of Product Lines. In 22nd International Conference on Software Engineering, 2000, pp.548-557.

S12 Geppert, B., Weiss, D.M. Goal-Oriented Assessment of Product-Line Domains. International Software Metrics Symposium (METRICS'03), 2003.

S13 Hwan, S., Kim, J., Kim, J. An Elicitation Approach of Measurement Indicator Based on Product Line Context. Proceedings of the Fourth International Conference on Software Engineering Research, 2006,

S14 Inoki, M., Fukazawa, Y.: Software Product Line Evolution Method Based on Kaizen Approach. In 22nd Annual ACM Symposium on Applied Computing, Korea, 2007.

S15 Johansson, E., Höst, R.: Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations. In 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, pp.249-254, 2002.

S16 Kim, T., Ko, I.Y., Kang, S.W., and Lee, D.H. "Extending ATAM to assess product line architecture". In 8th IEEE International Conference on Computer and Information Technology, pp. 790-797, 2008.

S17 Land, R., Alvaro, A., Crnkovic, I. Towards Efficient Software Component Evaluation: An Examination of Component Selection and Certification. 34th Euromicro Conference Software Engineering and Advanced Applications, 2008.

S18 Lin, Y., Ye, H., and Tang, J. Measurement of the Complexity of Variation Points in Software Product Lines. World Congress on Software Engineering, 2009.

S19 Lutz, R.R., Gannod, G.C. Analysis of a software product line architecture: an experience report. Journal of Systems and Software 66 (2003) 253-267.

S20 Mellado, D., Rodríguez, J., Fernández-Medina, E., and Piattini, M. Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering. International Conference on Availability, Reliability and Security, 2009.

S21 Meyer, M. H., and Dalal, D. Managing platform architectures and manufacturing processes for non assembled products. Journal of Product Innovation Management. Volume 19, Issue 4, pages 277–293, July 2002.

S22 Misic, V. B. Measuring the Coherence of SoftwareProduct Line Architectures. technical report TR 06/03, Department of Computer Science, University of Manitoba, June 2006.

S23 Mustapic, G., Andersson, J., Norström, C., and Wall, A. A Dependable Open Platform for Industrial Robotics – A Case Study. In Architecting Dependable Systems II. Lecture Notes in Computer Science, 2004, Volume 3069/2004, 307-329

S24 Needham, D., Jones, S.: A Software Fault Tree Metric. In 22nd International Conference on Software Maintenance (ICSM), Philadelphia, Pennsylvania, USA, 2006.

S25 Rahman, A.: Metrics for the Structural Assessment of Product Line Architecture. Master Tesis on Software Engineering, Thesis no. MSE-2004:24. School of Engineering, Blekinge Institute of Technology, Sweden, 2004.

S26 Sethi, K. Cai, Y., Wong, S., Garcia, A., Sant'Anna, C. From Retrospect to Prospect: Assessing Modularity and Stability from Software Architecture. Joint Working IEEE/IFIP Conference on Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009.

S27 Shaik I., Abdelmoez, W,. Gunnalan, R., Shereshevsky , M., Zeid, A., Ammar, H.H., Mili, A., and Fuhrman, C. Change Propagation for Assessing Design Quality of Software Architectures. 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005.

S28 Siegmund, N., Rosenmüller, M., Kuhlemann, M., Kästner, C., Saake, G.: Measuring Non-functional Properties in Software Product Lines for Product Derivation. In 15th Asia-Pacific Software Engineering Conference, Beijing, China, 2008.

S29 Stoll, P., Bass, L., Golden, E., John, B.E. Supporting Usability in Product Line Architectures. In Software Product Lines Conference, 2009.

S30 Stuikys, V., Damasevicius, R. Measuring Complexity of Domain Models represented by Feature Diagrams. Information Technology and Control, 2009, Vol.38, No.3.

S31 Sun Her, J., Hyeok Kim, J., Hun Oh, S., Yul Rhew, S., Dong Kim, S.: A framework for evaluationg reusability of core asset in product line engineering. Information and Software Technology 49, pp. 740-760, 2007.

S32 Van der Hoek, A., Dincel, E., Medidović, N.: Using Services Utilization Metrics to Assess the Structure of Product Line Architectures. In 9th International Software Metrics Symposium (METRICS), Sydney, Australia, 2003.

S33 Wnuk, K., Regnell, B., Karlsson, L. What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting. In 17th IEEE International Requirements Engineering Conference, 2009.

S34 Yoshimura, K., Ganesan, D., Muthig, D.: Assessing Merge Potential of Existing Engine Control Systems into a Product Line. In International Workshop on Software Engineering for Automative Systems, Shangai, China, pp.61-67, 2006.

S35 Zhang, T., Deng, L., Wu, J., Zhou, Q., Ma, C.: Some Metrics for Accessing Quality of Product Line Architecture. In International Conference on Computer Science and Software Engineering (CSSE), Wuhan, China, pp.500-503, 2008.

Appendix B

Classification of software measures found in the systematic review.

Legend		
Measure	Name of measure	
Characteristic	F = Functional Suitability, R = Reliability, Pe = Performance efficiency, O = Operability, S = Security, C = Compatibility, M = Maintainability, T = Transferability	
Quality Attribute	Name: name of the quality attribute	
	Tp: Type of the attribute	I = Internal, E = External
Type of Measure	QN = Quantitative, QL = Qualitative	
	E = Exact, P = Probabilistic	
Life-cycle Phase	DE = Domain Engineering, DA = Domain Application	
	Req. = Requirements, Des. = Design, Real. = Realization, Test = Test, Evol. = Evolution	
Artifact evaluated	PLA = Product Line Architecture, Ass = Asset, Pro Arc = Product Architecture, Fin Pro = Final Product	
Other characteristics	C = Compositionality	
	V = Variability	
Validation	T = Theoretical validation, Met = Method	PB = Property-based approach, MT = Measurement-theory approach
	E = Empirical validation, Met. = Method	CS = Case Studies, S = Surveys, CE = Controlled Experiments
Tool	x = Some tool exists	
Actual Usage	A = Academic, I = Industrial	
Reference	Paper reference from Appendix A	

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Binary Size	M	Binary Size	I	D	QN	E	DA Test		x		x			N	-	N	-	x	x	A	S28
Cyclomatic Complexity	M	Complexity of source code	E	D	QN	E	DA Test		x		x			N	-	N	-	x	x	A	S28
Performance	P	Performance	E	D	QN	E	DA Test				x			N	-	N	-	x	x	A	S28
Class AlternativeOR	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Class AlternativeXOR	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
Class Mandatory	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassNumVariantsAltOR	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassNumVariantsAltXOR	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassNumVariantsMandatory	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassNumVariantsOptional	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassOptional	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotal PLVariabilities	M	Complexity of an architecture (PLA)	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotal AlternativeOR	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotal AlternativeXOR	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotal Mandatory	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotalOptional	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotal Variabilities	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassTotalVP	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
ClassVP	M	Complexity of a class	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
Component TotalVariabilities	M	Complexity of a component diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Component Variable	M	Complexity of a component	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
CompPL	M	Complexity of an architecture (PLA)	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
CompVariability	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
CompVariant	M	Complexity of a class	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
CompVariantVP	M	Complexity of a class	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
CompVP	M	Complexity of a class	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
nVariants	M	Complexity of a class diagram	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
nVP	M	Complexity of a class diagram	I	B	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
PLTotalVariability	M	Complexity of an architecture (PLA)	I	D	QN	E	DE Des.	x				x	x	N	-	N	-	x	x	A	S04
UseCase AlternativeOR	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCase AlternativeXOR	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseMandatory	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseNum VariantsAltOR	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCase NumVariants AltXOR	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseNum VariantsMandatory	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseNum VariantsOptional	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
UseCaseOptional	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotal AlternativeOR	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotal AlternativeXOR	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotal Mandatory	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotal Optional	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotalPL Variabilities	M	Complexity of an architecture (PLA)	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotal Variabilities	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseTotalVP	M	Complexity of a use case diagram	I	D	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
UseCaseVP	M	Complexity of a use case	I	B	QN	E	DE Req.	x				x	x	N	-	N	-	x	x	A	S04
Maintainability Index (MI) (for a product)	M	Maintainability Index (MI) of a product	I	D	QN	E	DA Real.				x		x	N	-	N	-	x	x	A	S03
Maintainability Index (MI) (of a feature)	M	Maintainability Index (MI) of a feature	I	D	QN	E	DA Real.	x			x		x	N	-	N	-	x	x	A	S03
Maintainability Index (MI) (of the architecture)	M	Maintainability Index (MI) of the architecture	I	D	QN	E	DA Real.	x			x		x	N	-	N	-	x	x	A	S03
Architecture Variability (AV)	M	Variability of PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Component Reuse Rate (CRR)	M	Component reuse	I	D	QN	E	DA Des.			x			x	N	-	N	-	x	x	A	S35

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Exterior information flow complexity (EIFC)	M	Complexity of interaction of the ports of components and the roles of connectors	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Interior information flow complexity (IIFC)	M	Complexity of interior information flow arc of component and connector	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
PLA-IFG Cyclomatic Complexity (PCC)	M	Flow structure attribute of PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
PLA-IFG information flow complexity	M	Total complexity of the PLA partially from different point of view	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
PLA-IFG total complexity (PTC)	M	Total complexity of the PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
PLA-IFG vertex complexity (PVC)	M	Total complexity of the PLA partially from some point of view	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Reuse Benefit Rate (RBR)	M	Reuse Benefit	I	D	QN	E	DE Real.		x				x	N	-	N	-	x	x	A	S35
Strong Coupling Coefficient (SCC)	M	Strong Coupling of PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Structure Similarity Coefficient (SSC)	M	Similarity of PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Structure Variability Coefficient (SVC)	M	Structure Variability Coefficient (SVC)	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Variability Points Number (VP)	M	Variability of PLA	I	B	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Weak Coupling Coefficient (WCC)	M	Weak Coupling of PLA	I	D	QN	E	DE Des.	x					x	N	-	N	-	x	x	A	S35
Architectural Commonality (AC)	M	Architectural commonality	I	D	QN	E	DE Des.	x					x	Y	PB	N	-			A	S31
Component Compliance (CC)	M	Component replaceability	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31
Coverage of Variability (CV)	M	Variability richness	I	D	QN	E	DE Des.	x	x				x	Y	PB	N	-			A	S31
Cumulative Applicability (CA)	M	Applicability	I	D	QN	E	DE Des.	x	x				x	Y	PB	N	-			A	S31
Effectiveness of Tailoring (ET)	M	Effectiveness of Tailoring	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31
Functional Coverage (FC)	M	Functional commonality	I	D	QN	E	DE Des.	x					x	Y	PB	N	-			A	S31
Non-Functional Commonality (NFC)	M	Non-functional commonality	I	D	QN	E	DE Des.	x					x	Y	PB	N	-			A	S31
Non-functional Coverage (NC)	M	Non-functional features commonality that are not covered	I	D	QN	E	DE Des.	x					x	Y	PB	N	-			A	S31
Overall Understandability (OU)	O	Understandability	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31
Reusability (RE)	M	Reusability of a core asset	I	D	QN	E	DE Des.	x	x				x	Y	PB	N	-			A	S31
Tailorability (TL)	M	Tailorability	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31
Tailorability of Closed variability (TC)	M	Tailorability of Closed variability	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31
Tailorability of Open variability (TO)	M	Tailorability of Open variability	I	D	QN	E	DE Des.		x				x	Y	PB	N	-			A	S31

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Cognitive Complexity of a Feature Model (I)	M	Cognitive Complexity of a Feature Model	I	B	QN	E	DE Des.	x					x	N	-	N	-	x		A	S30
Cognitive Complexity of a Feature Model (II)	M	Cognitive Complexity of a Feature Model	I	B	QN	E	DE Des.	x					x	N	-	N	-	x		A	S30
Compound Complexity of a Feature Diagram	M	Compound Complexity of a Feature Diagram	I	D	QN	E	DE Des.	x					x	N	-	N	-	x		A	S30
Structural Complexity of a Feature Model	M	Structural Complexity of a Feature Model	I	D	QN	E	DE Des.	x					x	N	-	N	-	x		A	S30
Efficiency for any single derivate product	P	Final product efficiency	I	D	QN	E	Evol.				x			N	-	Y	CS			A	S21
Efficiency for the entire stream of derivative products based on a common PLA	P	PLA efficiency	I	D	QN	E	Evol.	x			x			N	-	Y	CS			A	S21
Reuse for any single product	M	Final product reusability	I	D	QN	E	Evol.	x			x			N	-	Y	CS			A	S21
Reuse for the entire stream of derivative products based on a common platform architecture	M	PLA reusability	I	D	QN	E	Evol.	x			x			N	-	Y	CS			A	S21
Adjust product line growth	M	Effort	I	D	QN	E	Evol.	x	x					N	-	N	-	x		I	S02
Changes on product line	M	Effort	I	D	QN	E	Evol.	x			x			N	-	N	-	x		I	S02
Code churn (changes in product line layer)	M	Effort	I	B	QN	E	Evol.	x						N	-	N	-	x		I	S02

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Code churn (for product layer)	M	Effort	I	D	QN	E	Evol.				x			N	-	N	-	x		I	S02
Code churn on product line	M	Effort	I	D	QN	E	Evol.	x			x			N	-	N	-	x		I	S02
Efficiency	M	Efficiency	I	D	QN	E	Evol.	x			x			N	-	N	-	x		I	S02
Impact of change	M	Effort	I	D	QN	E	Evol.	x						N	-	N	-	x		I	S02
Number of modules	M	Size	I	B	QN	E	Evol.	x						N	-	N	-	x		I	S02
Product line growth	M	Size	I	D	QN	E	Evol.	x						N	-	N	-	x		I	S02
Size of code in the product line	M	Size	I	B	QN	E	Evol.	x						N	-	N	-	x		I	S02
Size of product code	M	Size	I	B	QN	E	Evol.				x			N	-	N	-	x		I	S02
Source of change	M	Effort	I	D	QN	E	Evol.	x	x		x			N	-	N	-	x		I	S02
Software Fault Tree Metric	S	Security of product architecture	I	D	QN	E	DA Des.			x		x	x	N	-	N	-	x		I	S24
Average of number of age of a fault	M	Maturity	E	D	QN	E	DE Test		x					N	-	Y	CS			A	S25
Average of number of days to close a fault	M	Maturity	E	D	QN	E	DE Test		x					N	-	Y	CS			A	S25
Configuration Complexity Metric	M	Configuration Complexity	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Constraint Complexity Metric	M	Constraint complexity	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Interface Complexity Metric	M	Interface Complexity of a sofware component	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Modularity	M	Modularity of the architecture	I	D	QN	E	DE Des.	x						N	-	Y	CS			A	S25
Number of closed faults	M	Maturity	E	B	QN	E	DE Test		x					N	-	Y	CS			A	S25

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Number of constraints on sequence of interface invocations	M	Interface Constraints Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of different configurations a component can operate in	M	Interface Packaging and Configurations Complexity	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of distinct range constraints on properties	M	Interface Constraints Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of events	M	Interface Signature Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of faults in requirements and design	M	Maturity	E	B	QN	E	DE Test		x					N	-	Y	CS			A	S25
Number of open faults	M	Maturity	E	B	QN	E	DE Test		x					N	-	Y	CS			A	S25
Number of post-conditions	M	Interface Constraints Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of pre-conditions	M	Interface Constraints Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of properties	M	Interface Signature Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of readable properties	M	Observability	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of services	M	Interface Signature Complexity	I	B	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Number of writable properties	M	Customizability	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Avg. Nº of days a fault remains	M	Maturity	E	D	QN	E	DE Test		x					N	-	Y	CS			A	S25

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Overall Interface Complexity of a Component Metric	M	Interface Complexity of a Component	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Ratio de servicios provistos sin parámetros de un componente (SCp(c))	M	Self Completeness (Degree of independence in term of the functionality that it provides)	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Ratio de servicios provistos sin valor de retorno de un componente (SCr(c))	M	Self Completeness (Degree of independence in term of the functionality that it provides)	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Reusability of architecture	M	Reusability of architecture	I	D	QN	E	DE Des.	x	x					N	-	Y	CS			A	S25
Signature Complexity Metric	M	Signature Complexity	I	D	QN	E	DE Des.		x					N	-	Y	CS			A	S25
Compound PSU (CPSU)	M	Internal cohesion	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Compound RSU (CRSU)	M	Internal cohesion	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Internal cohesion of architecture	M	Internal cohesion	I	D	QL	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Number of services provided by component x that is actually used by other components in the architecture (Pactual(x))	M	Structural soundness	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Number of services required by component x that is actually used by other components in the architecture (Ractual(x))	M	Structural soundness	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Provided Service Utilization (PSU)	M	Structural soundness	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Required Service Utilization (RSU)	M	Structural soundness	I	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S32
Total number of services provided by component x (Ptotal(x))	M	Structural soundness	I	B	QN	E	DE Des.		x			x		N	-	N	-	x		A	S32
Total number of services required by component x (Rtotal(x))	M	Structural soundness	I	B	QN	E	DE Des.		x			x		N	-	N	-	x		A	S32
Error Propagation	R	Error propagation	I	D	QN	P	DE Des.	x				x		N	-	Y	CS		x	A	S01
Average PSU	M	Cohesion of a complex component	I	D	QN	E	DE Des.		x			x		N	-	N	-	x		A	S06
Average PSU per Product Family Architecture	M	Average PSU per Product Family Architecture	I	D	QN	E	DE Des.	x				x		N	-	N	-	x		A	S06
Average RSU	M	Cohesion of a complex component	I	D	QN	E	DE Des.		x			x		N	-	N	-	x		A	S06
Average RSU per Product Family Architecture	M	Average RSU per Product Family Architecture	I	D	QN	E	DE Des.	x				x		N	-	N	-	x		A	S06

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Number of days from the beginning of the project until a feature was included	P	Time when a feature was included into the scope of the project	I	B	QN	E	DE Des.	x						N	-	Y	CS		x	A	S33
Number of days needed to make a final decision about feature exclusion	P	Number of scope decision per feature	I	B	QN	E	DE Des.	x						N	-	Y	CS		x	A	S33
Number of scope chages for non-survivors needed to mrevoe them from the scope	P	Number of scope decisions per feature	I	B	QN	E	DE Des.	x						N	-	Y	CS		x	A	S33
Number of scope exclusions at the timestamp	P	Size and direction of scope changes over time	I	B	QN	E	DE Des.	x						N	-	Y	CS		x	A	S33
Number of scope inclusions at the timestamp	P	Size and direction of scope changes over	I	B	QN	E	DE Des.	x						N	-	Y	CS		x	A	S33
Reason for scope exclusions	P	Rationale for removing features from the scope	I	B	QL	E	DE Des.	x						N	-	Y	CS			A	S33
Number of kinds of core assets	F	Level of Coverage	I	B	QN	E	DE Des.	x	x					N	-	N	-	x		A	S14
Total number of core assets	F	Level of Coverage	I	B	QN	E	DE Des.	x	x					N	-	N	-	x		A	S14
Clone Coverage	M	Similarity	I	D	QN	E	DE Real.		x					N	-	N	-	x		I	S34
Tracking Degradation	M	Tracking Degradation	E	D	QN	E	DA Test	x	x	x	x			N	-	N	-	x		I	S15
Coherence of a component	M	Coherence	E	D	QN	E	DE Des.	x	x			x		N	-	N	-	x		A	S22

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Cost of testing a single product	M	Cost of testing a single product	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Percentage of non-generic reusable components per product	M	Percentage of non-generic reusable components per product	E	D	QN	E	DE Des.			x		x		N	-	N	-	x	x	I	S10
Percentage of product-specific components per product	M	Percentage of product-specific components per product	E	D	QN	E	DE Des.			x		x		N	-	N	-	x	x	I	S10
Percentage of slightly generic reusable components per product	M	Percentage of slightly generic reusable components per product	E	D	QN	E	DE Des.			x		x		N	-	N	-	x	x	I	S10
Percentage of very generic reusable components per product	M	Percentage of very generic reusable components per product	E	D	QN	E	DE Des.			x		x		N	-	N	-	x	x	I	S10
Relative cost to test a non-generic component	M	Relative cost to test a non-generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Relative cost to test a slightly generic component	M	Relative cost to test a slightly generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Relative cost to test a very generic component	M	Relative cost to test a very generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Relative cost to test adaptations to a non-generic component	M	Relative cost to test adaptations to a non-generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10

Measure	Cha-racte-ristic	Quality Attribute		Type Meas.	Result of measure		Life-cycle Phase	Artifact evaluated				Other Charact.		Validation					Tool	Actual Usage	Refe-rence
		Name	Tp					PLA	Ass	Pro Arc	Fin Pro	C	V	T	Met	E	Met	Not Valid			
Relative cost to test adaptations to a slightly generic component	M	Relative cost to test adaptations to a slightly generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Relative cost to test adaptations to a very generic component	M	Relative cost to test adaptations to a very generic component	E	D	QN	E	DE Des.		x			x		N	-	N	-	x	x	I	S10
Variation Rank of a variant	M	Variability of a variant	I	D	QN	E	DE Des.	x					x	N	-	N	-	x		A	S18
Variation Rank of a variation point	M	Variability of a variation point	I	D	QN	E	DE Des.	x					x	N	-	N	-	x		A	S18
Change Propagation Coefficient (CPC)	M	Potencial of an architecture to insulate its components from each other's changes	I	D	QN	E	DE Des.	x	x			x		N	-	Y	CS		x	A	S27
Change propagation probability (CP)	M	Change propagation probability on architecture	I	D	QN	P	DE Des.	x	x			x		N	-	Y	CS		x	A	S27
Concern Overlap	M	Interaction between two concerns	I	D	QN	E	DE Des.	x						N	-	Y	CE		x	A	S26
Decision Volatility	M	Stability of a decision	I	D	QN	E	DE Des.	x						N	-	Y	CE		x	A	S26
Design Volatility	M	Stability of all decisions	I	D	QN	E	DE Des.	x						N	-	Y	CE		x	A	S26
Independence Level	M	Independece of the system	I	D	QN	E	DE Des.	x						N	-	Y	CE		x	A	S26
Binary Size	M	Binary Size	I	D	QN	E	DA Test		x		x			N	-	N	-	x	x	A	S28

Appendix C

List of quality attributes for software product lines found in the systematic review

Quality attribute	Number of occurrences	Number of Measures
Accuracy	1	0
Active domain	1	0
Applicability	1	1
Architectural commonality	1	1
Architecture compliance	1	0
Atomicity	1	0
Availability	3	0
Average PSU per Product Family Architecture	1	1
Average RSU per Product Family Architecture	1	1
Behaviour	1	0
Beneficio de la reusabilidad	1	0
Binary Size	1	1
Change propagation probability on architecture	1	1
Clarity of the reasons for scope decisions	1	0
Cognitive Complexity of a Feature Model	2	2
Coherence	1	1
Cohesion of a complex component	2	2
Commercial efficiency	1	0
Complexity	1	0
Complexity of a class	12	12
Complexity of a class diagram	9	9
Complexity of a component	1	1
Complexity of a component diagram	1	1
Complexity of a use case	9	9
Complexity of a use case diagram	6	6
Complexity of an architecture (PLA)	4	4
Complexity of interaction of the ports of components and the roles of connectors	1	1
Complexity of interior information flow arc of component and connector	1	1
Complexity of source code	1	1
Compliance	1	0
Component replaceability	1	1
Component reuse	1	1
Composability	1	0
Compound Complexity of a Feature Diagram	1	1
Confidentiality	1	0
Configurability	1	0
Configuration Complexity	1	1

Quality attribute	Number of occurrences	Number of Measures
Consistency	1	0
Constraint complexity	1	1
Cost for production of core assets	1	0
Cost of testing a single product	1	1
Cost of use of core assets	1	0
Customer satisfaction	1	0
Customizability	1	1
Defect density of applied artifacts	1	0
Defect density of core assets	1	0
Defect density trend	1	0
Deletions	2	0
Direct product cost	1	0
Durability	1	0
Effect to investment	1	0
Effectiveness of Tailoring	1	1
Efficiency	2	1
Effort	7	7
EMC	1	0
Energy efficiency	1	0
Error propagation	1	1
Evolution	1	0
Extensibility	5	0
Final product efficiency	1	1
Final product reusability	1	1
Flexibility	1	0
Flow structure attribute of PLA	1	1
Functional commonality	1	1
Functional requirements	1	0
Independece of the system	1	1
Independent domain	1	0
Infraestructure production cost	1	0
Integrity	2	0
Interaction between two concerns	1	1
Interface Complexity of a Component	1	1
Interface Complexity of a sofware component	1	1
Interface Constraints Complexity	4	4
Interface Packaging and Configurations Complexity	1	1
Interface Signature Complexity	3	3
Internal cohesion	3	3
Interoperability	2	0
Isolation	1	0

Quality attribute	Number of occurrences	Number of Measures
Level of Coverage	2	2
Maintainability	2	0
Maintainability Index (MI) of a feature	1	1
Maintainability Index (MI) of a product	1	1
Maintainability Index (MI) of the architecture	1	1
Market feature application scope	1	0
Market satisfaction	1	0
Maturity	6	6
Mission focus	2	0
Modifiability	1	0
Modularity of the architecture	1	1
Non-functional commonality	1	1
Non-functional features commonality that are not covered	1	1
Number of scope decision per feature	1	1
Number of scope decisions per feature	1	1
Observability	1	1
Operational time during parking	1	0
Percentage of non-generic reusable components per product	1	1
Percentage of product-specific components per product	1	1
Percentage of slightly generic reusable components per product	1	1
Percentage of very generic reusable components per product	1	1
Performance	3	1
Physical fitness	1	0
Physical weight of the system	1	0
PLA efficiency	1	1
PLA reusability	1	1
Portability	4	0
Potencial of an architecture to insulate its components from each other's changes	1	1
Power consumption during normal operation	1	0
Process compliance	3	0
Produceability	1	0
Productivity	1	0
Quality of core assets	1	0
Rationale for removing features from the scope	1	1
Relative cost to test a non-generic component	1	1
Relative cost to test a slightly generic component	1	1
Relative cost to test a very generic component	1	1
Relative cost to test adaptations to a non-generic component	1	1
Relative cost to test adaptations to a slightly generic component	1	1
Relative cost to test adaptations to a very generic component	1	1
Reliability	3	0

Quality attribute	Number of occurrences	Number of Measures
Responsiveness of the communication networks	1	0
Restructuring	2	0
Reusability	1	0
Reusability of a core asset	1	1
Reusability of architecture	1	1
Reuse rate	2	0
Revenue-producing domain	1	0
Robustness	1	0
Safety	5	0
Scalability	3	0
Schedule difference	1	0
Security	1	0
Security of product architecture	1	1
Self Completeness (Degree of independence in term of the functionality that it provides)	2	2
Serviceability	1	0
Signature Complexity	1	1
Similarity	1	1
Similarity of PLA	1	1
Size	4	4
Size and direction of scope changes over time	2	2
Stability of a decision	1	1
Stability of all decisions	1	1
Stability of the scoping process	2	0
Strong Coupling of PLA	1	1
Structural Complexity of a Feature Model	1	1
Structural soundness	6	6
Structure Variability Coefficient (SVC)	1	1
Styling compatibility	1	0
Suitability	1	0
Tailorability	1	1
Tailorability of Closed variability	1	1
Tailorability of Open variability	1	1
Testability	1	0
The number and type of artifacts in asset library	1	0
The number of products (past, current, and future)	1	0
Throughput	1	0
Time distributed over cycle time activity	1	0
Time to market	2	0
Time when a feature was included into the scope of the project	1	1
Total complexity of the PLA	1	1

Quality attribute	Number of occurrences	Number of Measures
Total complexity of the PLA partially from different point of view	1	1
Total complexity of the PLA partially from some point of view	1	1
Total product development cost	1	0
Tracking Degradation	1	1
Understandability	1	1
Usability	2	0
Usefulness of core assets	1	0
Variability of a variant	1	1
Variability of a variation point	1	1
Variability of PLA	2	2
Variability richness	1	1
Viable domain	1	0
Volatility and dynamics of the scope decisions	2	0
Volatility of the scope decisions	1	0
Weak Coupling of PLA	1	1