Software Qual J (2011) 19:579–612
DOI 10.1007/s11219-010-9127-2

Assessing the maintainability of software product line feature models using structural metrics

Ebrahim Bagheri · Dragan Gasevic

Published online: 1 January 2011

Abstract A software product line is a unified representation of a set of conceptually similar software systems that share many common features and satisfy the requirements of a particular domain. Within the context of software product lines, feature models are tree-like structures that are widely used for modeling and representing the inherent commonality and variability of software product lines. Given the fact that many different software systems can be spawned from a single software product line, it can be anticipated that a low-quality design can ripple through to many spawned software systems. Therefore, the need for early indicators of external quality attributes is recognized in order to avoid the implications of defective and low-quality design during the late stages of production. In this paper, we propose a set of structural metrics for software product line feature models and theoretically validate them using valid measurement-theoretic principles. Further, we investigate through controlled experimentation whether these structural metrics can be good predictors (early indicators) of the three main subcharacteristics of maintainability: *analyzability, changeability*, and *understandability*. More specifically, a four-step analysis is conducted: (1) investigating whether feature model structural metrics are correlated with feature model maintainability through the employment of classical statistical correlation techniques; (2) understanding how well each of the structural metrics can serve as discriminatory references for maintainability; (3) identifying the sufficient set of structural metrics for evaluating each of the subcharacteristics of maintainability; and (4) evaluating how well different prediction models based on the proposed structural metrics can perform in indicating the maintainability of a feature model. Results obtained from the controlled experiment support the idea that useful prediction models can be built for the purpose of evaluating feature model maintainability using early structural metrics. Some of the

E. Bagheri (✉) · D. Gasevic
Athabasca University, Athabasca, Canada
e-mail: ebagheri@athabascau.ca

D. Gasevic
e-mail: dragang@athabascau.ca

E. Bagheri
National Research Council, Ottawa, ON, Canada

structural metrics show significant correlation with the subjective perception of the subjects about the maintainability of the feature models.

Keywords Software product line · Feature model · Quality attributes · Maintainability · Structural complexity · Controlled experimentation · Software prediction model

1 Introduction

In the realm of object-oriented software development, it has been widely recognized that in order to be able to design high-quality software systems, the developers will need to focus on developing high-quality conceptual models of the intended systems in the early stages of development (Korson and McGregor 1990; Siau and Tan 2005). Therefore, the emphasis should be on maintaining the quality of the designed conceptual models of the software rather than trying to enforce quality at the late stages of development, e.g., coding stages. In such a model-driven development paradigm (Kleppe et al. 2003), the quality of models is extremely important because it will directly impact the quality of the final software product (Selic 2006). A fortiori, the enforcement of quality in software product line conceptual models is necessary. This is due to the fact that software product lines serve as the founding basis for the development of families of software systems that have common functionality and behavior (Pohl et al. 2005); hence, while a low-quality model of a single software system will at most affect the quality of that software and perhaps its later versions, in software product lines this deficiency will translate into many different low-quality or even erroneous software systems that are spawned from the product line (Pohl and Metzger 2006).

Given that software product lines are gaining interest both in industrial environments as illustrated by success reports from Nokia, Boeing and Toshiba among others (Weiss et al. 2006; McGregor et al. 2010) and also in academia, it is important to design methodologies and processes for creating high-quality software product line conceptual models, which are most often in the form of *software product line feature models* (Lee et al. 2002; Kang et al. 1990). At this time, no well-established and widely used methodology for such purpose exists. One of the reasons for this may be the lack of appropriate mechanisms for measuring the properties of software product lines. As put by Galileo: *what is not measurable make measurable*, which is an indication that measurement at least serves to understand, control, and improve a process that is frequently performed. This fact was also shown under the *Hawthorne effect* in organizational theory which states that *what is measured is improved* (Scott 1998). Many software engineering researchers have used measurement as means of improving software quality (Fenton 1994; Jones 2008). Therefore, the objective of our work is to define a set of structural metrics for software product line feature models for measuring the quality of the designed models.

In practice, the availability of early structural metrics, such as degree of cohesion and coupling in object-oriented programs (Hitz and Montazeri 1995), can allow for a quantitative comparison of design alternatives and hence provide for an objective evaluation and comparison between them (Fenton and Neil 1999). The metrics would also serve to improve the quality of the resulting software products by helping to predict the possible quality of the final system and improve the resource allocation process based on these predictions (Bansiya and Davis 2002).

With respect to software artifacts, quality attributes can be categorized into two classes: *internal quality attributes* and *external quality attributes* (Barbacci et al. 1995). Internal

quality attributes are those that can be directly measured purely on the basis of product features such as size, length, or complexity. On the other hand, external quality attributes, e.g., efficiency, reliability, and maintainability, are those quality attributes that can only be measured with respect to how a software system relates with its environment and can hence only be measured once the software system is fully developed and deployed. *Maintainability* as one of such external quality attributes is concerned with evaluating how well the developed software models can be understood, changed, and analyzed (Genero et al. 2001). As will be discussed in the later sections of this paper and used throughout, we capture maintainability as the subjective perception of domain analysts with regard to the subcharacteristics of this external quality attribute.

Since external quality attributes are hard to objectively evaluate early in the software development process, an indirect measurement based on internal quality attributes is often devised. Research in the field of empirical software engineering has already shown that internal quality attributes can be appropriate determinants for external quality attributes (Manso et al. 2010; Briand et al. 2000). Based on such observations, the focus of our work in this paper is on devising appropriate internal quality attributes in terms of software product line feature model structural metrics and using them as early indicators of *maintainability* which is an important external quality attribute. We view maintainability as a composition of three subcharacteristics, namely *analyzability*, *changeability*, and *understandability* (Al-Kilidar et al. 2005). As proposed in the ISO 9126 standard for software quality (ISO 2001), these three concepts are interpreted as follows:

- *analyzability* is the capability of the conceptual model of a software system to be diagnosed for deficiency;
- *changeability* is the possibility and ease of change in a model when modifications are necessary;
- *understandability* is the prospect and likelihood of the software system model to be understood and comprehended by its users or other model designers.

We analyze both the empirical and theoretical views of the developed structural metrics for software product line feature models. Our aim is to show that the defined metrics are valid from a measurement-theoretic perspective (*construct validity*) in both the *property-based measurement framework* (Briand et al. 1996) and the *DISTANCE framework* (Poels and Dedene 2000). These two frameworks are among the important evaluation frameworks for evaluating software metrics from a measurement-theoretic view. Empirical experiments are also performed to analyze the relevance of the defined metrics to be used as early indicators of maintainability. Overall, the objective of our work is twofold: (1) to devise a set of feature model structural metrics that serve as internal quality attributes and to show that they respect the properties required within the framework of measurement theory and (2) to investigate and analyze whether these structural metrics are good early indicators of maintainability. If the proposed structural metrics are shown to have meaningful correlation with maintainability and its subcharacteristics, they would allow software product line designers to make better and more informed decisions earlier in the software development process.

The rest of this paper is organized as follows: Software product lines and its related conceptual modeling formalism, feature modeling, is introduced in Sect. 2. Section 3 describes the proposed software product line structural quality framework. The setup and the structure of the experiment is presented in Sect. 4, which is followed by the explanation of the employed analysis techniques used for interpreting the collected data is described in Sect. 5 The results of the performed analysis are presented in Sect. 6 Some general

discussions and related work are provided in Sects. 7 and 8, respectively. The paper is then concluded in Sect. 9.

2 Software product line concepts

Software product line engineering is concerned with capturing the commonalities, universal and shared attributes of a set of software-intensive applications for a specific problem domain (Pohl et al. 2005). It allows for the rapid development of variants of a domain-specific application through various configurations of a common set of reusable assets often known as *core assets*, which support the management of commonality as well as variability. On the one hand, *commonality* is supported by providing domain analysts with both the ability and the required tools for capturing all of the shared conceptual information within the applications of a given domain. On the other hand, *variability* is addressed by allowing the domain analysts to include application-specific attributes and features within their unified model but at the same time, restrict their use; this way, commonality and variability are handled simultaneously. In the context of software product lines, *feature modeling* is one of the important techniques for modeling the attributes of a family of systems (Lee et al. 2002; Czarnecki et al. 2005). This modeling language is important in that it provides means for capturing variability in software product lines (Babar et al. 2010).

2.1 Feature models

Features are important distinguishing aspects, qualities, or characteristics of a family of systems (Lee et al. 2002; Kang et al. 1990). They are used for depicting the shared structure and behavior of a set of similar systems. To form a product family, all the various features of a set of similar or related systems are composed into a feature model. A feature model is a means for representing the possible configuration space of all the products of a system product family in terms of its features. For this reason, it is important that feature models be able to capture variability and commonality between the features of the different applications available in a given domain. As we will see in the following paragraphs, feature models provide suitable means for modeling commonality, by allowing the domain modelers to form a common feature model representation for multiple applications, as well as variability by providing means to capture competing features of different applications under one unified umbrella. An example of capturing commonality is when a similar feature, which exists in multiple applications, is represented as a unique feature in the overall domain representation, while an example of variability is when one notion is viewed differently by separate applications and is therefore modeled using competing features.

Feature models can be represented both formally and graphically; however, the graphical notation depicted through a tree structure is more favored due to its visual appeal and easier understanding. More specifically, graphical feature models are in the form of a tree whose root node represents a domain concept, e.g., a domain application, and the other nodes and leaves illustrate the features. In this context, a feature is a concept or property related to a user-visible functional or non-functional requirement, e.g., domain application task, modeled in a way to capture commonalities or possibly differentiate among product family variants (Tessier et al. 2005).

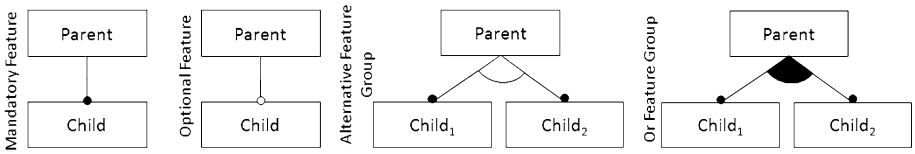

Fig. 1 The main graphical notations employed in feature modeling

In a feature model, features are hierarchically organized and can typically be classified as:

- *Mandatory*, the feature must be included in the description of its parent feature;
- *Optional*, the feature may or may not be included in its parent description given the situation;
- *Alternative feature group*, one and only one of the features from the feature group can be included in the parent description;
- *Or feature group*, one or more features from the feature group can be included in the description of the parent feature.

Figure 1 depicts the graphical notation of the feature relationships. The tree structure of feature models falls short at fully representing the complete set of mutual interdependencies of features; therefore, additional constraints are often added to feature models and are referred to as *Integrity Constraints (IC)*. The two most widely used integrity constraints are as follows:

- *Includes*, the presence of a given feature (or set of features) requires the *existence* of another feature (or set of features);
- *Excludes*, the presence of a given feature (or set of features) requires the *elimination* of another feature (or set of features).

2.2 The graph product line feature model

The Graph Product Line (GPL) (Lopez-Herrejon and Batory 2001), which is designed with the ambition to be a standard benchmark for evaluating feature modeling techniques shown in Fig. 2, is the standard feature model in the software product line community that covers the classical set of applications of graphs in the domain of Computer Science. The intention is to be able to develop configurations of GPL, which are able to address different problems. For instance, GPL can be configured to perform several graph search algorithms over a directed or undirected graph structure.

As it can be seen, GPL consists of three main features: (1) Graph Type: features for defining the structural representation of a graph; (2) Search: traversal algorithms in the form of features that allow for the navigation of a graph; (3) Algorithms: other useful algorithms that manipulate or analyze a given graph. Clearly, not all possible configurations of the features of GPL produce valid graph programs. For instance, a configuration of GPL that checks if a graph is strongly connected cannot be implemented on an undirected graph structure. Such restrictions are expressed in the form of integrity constraints. Some examples of these constraints are as follows:

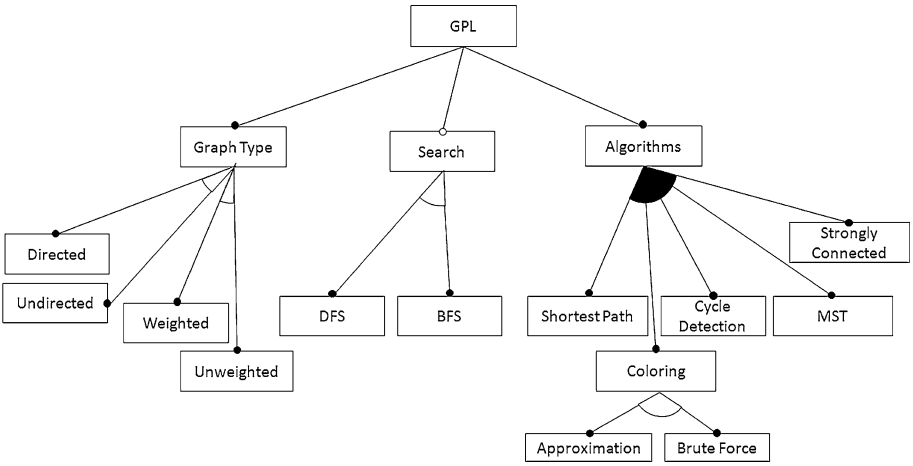

Fig. 2 The graph product line feature model

$$\mathtt{IC} = \begin{cases} \text{Cycle Detection } \textsc{excludes} \text{ BFS;} \\ \text{Cycle Detection } \textsc{includes} \text{ DFS;} \\ \text{Strongly Connected } \textsc{includes} \text{ DFS;} \\ \text{Strongly Connected } \textsc{includes} \text{ Directed;} \end{cases}$$

These integrity constraints ensure the correct composition of features in the various final software products developed from this feature model.

3 The quality framework

3.1 Considerations about metrics design

The underlying rationale for developing quantitative models for interrelating structural model properties and external quality attributes has been identified to be related to concept of *cognitive complexity* (Cant et al. 1994; Briand et al. 2001). Simon (1978) has extensively argued about the limitations of human cognitive capabilities and concluded that humans act rationally to the extent of their cognitive capability; therefore, they can lose track of what they are doing and make irrational decisions in complex and large environments. This also applies to software modelers who may become overwhelmed with the complexity of the domain that they are dealing with and therefore can introduce errors in the models being designed. In other words, cognitive complexity is the mental burden of the people that perform relevant operations on the software; hence, the larger and more complex the software system is, the higher its degree of cognitive complexity will be. High cognitive complexity of a software can impact its external quality attributes.

According to Briand et al.'s model (1999), referred to as *causal chain*, which is a basis for much empirical software quality metrics research, it is difficult to imagine what could be an alternative reason for cognitive complexity other than software structural properties. As shown in Fig. 3, the structural properties of a software system have impact on its cognitive complexity and can be considered as indications for external quality attributes. Software product line feature models are in fact forms of software system models, so it is

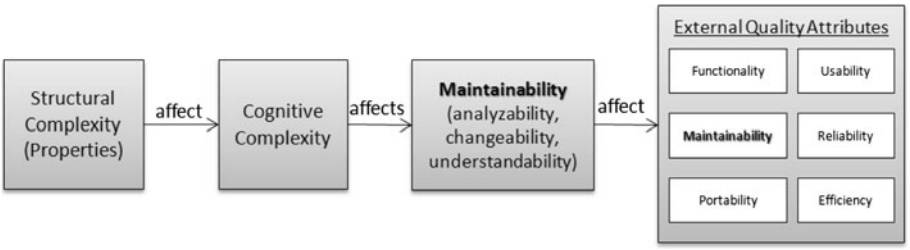

Fig. 3 The relationship between structural properties, cognitive complexity, and external quality attributes as described by Briand et al. (1999)

reasonable to hypothesize that they may have the same characteristics in terms of relationship between their cognitive complexity and their structural properties. It is important to study this relationship in order to be able to create appropriate early metrics for software product line feature models to serve as indicators of external quality attributes.

3.2 Proposed structural metrics

To the extent of our knowledge, only very few preliminary studies for defining suitable feature model structural metrics have been conducted (de Oliveira et al. 2008; Zhang et al. 2008). Even within these existing works, not much theoretical or empirical evaluation of the proposed metrics has been done. Given the fact that Fenton has formally proven that a single metric cannot capture all of the various aspects of complexity (Fenton 1994), in our current study we propose ten simple yet intuitive structural metrics as measurement references for software product line feature models. The main idea behind the design of these metrics has been *comprehensiveness* and *simplicity*. We have tried to cover as many structural characteristics of a feature model as possible. To achieve this, structural metrics proposed in the areas of object-oriented design, entity-relationship diagrams, and even business process models have been considered; the empirical results of which were taken into account as guidelines for designing the most useful set of structural metrics for our purpose (Genero et al. 2008; Cruz-Lemus et al. 2009, 2010; Serrano et al. 2008). In addition, based on *Ocam's Razor*, and also empirical results (e.g., although LOC is a naïve and simple measure, it performs well in some circumstances (Fenton and Neil 1999)) the design of simple metrics has been our secondary objective.

Table 1 provides the description of each of the proposed metrics. The metrics were selected based on a classification provided by Briand et al. (1996) for object-oriented design models. These authors provided five types of metrics, i.e., *size, length, complexity, coupling*, and *cohesion*. It is clear that the last two metric types (cohesion and coupling) are only relevant for modular systems and are not applicable to our domain. So, we have attempted to define metrics for each of the other types: size (NF, NTop, NLeaf), length (DT), and complexity (CC, CTC, RoV, CoC, FoC, NVC).

It should be noted that analogous to metric design for other software engineering discipline, this set of metrics may not be comprehensive and other consecutive research could further complete this proposed set by defining new metrics from other perspectives. Table 2 depicts the calculated values of each of these metrics for the graph product line shown in Fig. 2 in order to facilitate the understanding of the metrics and to also help in

Table 1 Measures for software product line feature models

Measure type	Measure name	Measure definition
Size measures	Number of features (NF)	The total number of features that are present in a feature model. This includes both leaf and parent features. NF counts all of the nodes in the feature model tree
	Number of top features (NTop)	The number of features that are first direct descendants of the feature model root. In other words, the number of nodes in depth one of the tree
	Number of leaf features (NLeaf)	The number of features with no children or further specializations. These correspond with the leafs of the feature model tree
Structural complexity measures	Cyclomatic complexity (CC)	The number of distinct cycles that can be found in the feature model. Since feature models are in the form of trees, no cycles can exist in a feature model; however, integrity constraints between the available features can cause cycles. It is simple to show that the number of distinct cycles and hence *cyclomatic complexity* of a feature model is equivalent to the number of integrity constraints of a feature model. This is due to the tree-like (cycleless graph) structure of feature models
	Cross-tree constraints (CTC)	The ratio of the number of unique features involved in the feature model integrity constraint over all of the number of features in the feature model. This measure represents the degree of involvement of features in the definition of the integrity constraints
	Ratio of variability (RoV)	The average branching factor of the parent features in the feature model. In other words, the average number of children of the nodes in the feature model tree
	Coefficient of connectivity—density (CoC)	The ratio of the number of edges over the number of features in a feature model. In graph theory, the *coefficient of connectivity* represents how well the graph components are connected
	Flexibility of configuration (FoC)	This is the ratio of the number of optional features over all of the available features in the feature model. The rationale behind this is that the more optional features exist in the feature model, the more choices are available for the designers to choose from while configuring the feature model
	Number of valid configurations (NVC)	The number of all possible and valid configurations that can be derived from the feature model in the face of its integrity constraints and tree structure
Length measure	Depth of tree (DT)	The length of the longest path from the feature model root to leaf features in the feature model

clearly showing how the metrics are calculated over this simple feature model. The GPL shown in Fig. 2 is accessible from SPLOT.[1]

3.3 Theoretical analysis of metrics

A software metric or generally any measure is a *homomorphism* from an empirical relational system to a numerical relational system (Finkelstein 2003); therefore, it is imperative

[1] http://splot-research.org/.

Table 2 Measures values for GPL feature model depicted in Fig. 2

Measure	Value
NF	17
NTop	3
NLeaf	12
CC	4
CTC	0.294
RoV	3.2
CoC	1.176
FoC	0.058
NVC	63
DT	4

that measures be theoretically analyzed within the framework of measurement theory. There are two main approaches for such theoretical analysis within software measurement literature: (1) frameworks directly based on measurement theory principles such as the DISTANCE framework (Poels and Dedene 2000) and (2) frameworks based on desirable properties (axioms) such as Briand et al.'s (1996) property-based measurement framework.

Frameworks such as DISTANCE (Poels and Dedene 2000) ensure that the metrics developed based on their guidelines are proven to be valid distance measures and that they can be used as *ratio scale* measurement instruments. On the other hand, in the property-based (axiomatic) approaches (Briand et al. 1996), instead of explicitly defining the empirical relational system, the desirable properties of the numerical relational system that need to be satisfied by the metrics are expressed. We examine the properties of our proposed metrics in the context of both of these frameworks.

In brief, the DISTANCE framework proposes a set of *mandatory* properties, i.e., *non-negativity, identity, symmetry*, and *triangular inequality* that need to be satisfied by any metric in order for it to be considered an acceptable measurement-theoretic metric. In addition, the property-based measurement framework provides a set of *desirable* properties for different metric types: size (*non-negativity, null value, additivity*), length (*non-negativity, null value, identity, monotonicity*), and complexity (*non-negativity, identity, symmetry, additivity, monotonicity*) and recommends that these properties are satisfied as much as possible. We investigate to what extent our proposed metrics are able to respect these properties.

The summary of the theoretical properties of our metrics obtained by following the steps proposed in (Poels and Dedene 1999) is shown in Table 3. As it can be seen, all of the introduced metrics respect the four mandatory properties required by the DISTANCE framework to form a valid *metric space*. Therefore, the important consequence of satisfying these four properties is that all of our proposed metrics are in *ratio scale* (Fenton and Pfleeger 1997), which means that they are theoretically valid software metrics.

Furthermore, Table 3 shows that other than the metrics in the complexity type, the rest (metrics for size and length) respect the properties defined for their respective types within the property-based measurement framework. As for the complexity metrics, they respect four out of the five desirable properties defined by the property-based measurement framework for complexity. They fail to respect the *additivity* property. We believe that although the defined complexity metrics do not exactly satisfy *additivity*, but their satisfaction of the *monotonicity* properties (*Non-increasing Monotonicity for Connected*

Table 3 Theoretical properties of the defined metrics

Properties	Size			Complexity						Length
	(NF	NTop	NLeaf)	(CC	CTC	RoV	CoC	FoC	NVC)	(DT)
Non-negativity†,‡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Null value†	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Symmetry†,‡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Non-increasing monotonicity (connected components)†	*NA*	*NA*	*NA*	✓	✓	✓	✓	✓	✓	✓
Identity‡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Non-decreasing monotonicity (disjoint components)†	*NA*	*NA*	*NA*	✓	✓	✓	✓	✓	✓	✓
Additivity†	✓	✓	✓	✓	×	×	×	×	×	*NA*
Triangular inequality‡	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

† and ‡ signify that the property is defined in the property-based measurement framework and the DISTANCE framework, respectively

Components and *Non-decreasing Monotonicity for Disjoint Components*) makes them acceptable for our purpose. This is because it is important to make sure that as a model becomes more (less) complex, the metrics measuring complexity will also reflect this increase (decrease) in complexity accordingly, and therefore, monotonicity is an essential property for complexity metrics. Hence, we ensured that our metrics conform to monotonicity. However, additivity requires if two models are merged, the complexity of the resulting model is equivalent to the sum of the complexity of the two source models. This is a strict monotonicity requirement and does not necessarily need to be respected.

As a matter of fact, additivity is only a special case of monotonicity, i.e., monotonicity is less strict than additivity. For instance, assuming that A is a more complex system than B, and μ is the measure of complexity: $\mu(A) = 4$, and $\mu(B) = 1$, an additive measure such as μ is required to satisfy $\mu(A + B) = 5$, but a monotone μ would only require $\mu(A + B) > 4$; therefore, monotonicity is less strict than additivity. It is believed that monotonicity as respected by our metrics is strict enough to show the complexity relations between feature models and a measure does not essentially need to be additive (while being monotone) to be a valid complexity measure.

It is worth noting that despite the fact that our complexity metrics do not satisfy additivity, they are still valid measurement-theoretic metrics as shown by satisfying the *mandatory* properties of the DISTANCE framework and only fail to respect one of the five *desirable* properties of complexity metrics proposed by the property-based measurement framework. Similar to this case, it has been shown in the literature that some very useful software metrics can only satisfy a subset and not all of the desirable properties of the property-based measurement framework (Briand et al. 1996) such as the widely used metrics proposed by Chidamber et al. (1994).

Now, given the introduction of a set of theoretically valid software metrics for software product line feature models, we need to empirically validate their usefulness for quality assessment in practice. In the remainder of this paper, we will use some advanced

statistical analysis techniques to evaluate the efficiency of these metrics for external quality attribute prediction and also to analyze the interaction between the proposed metrics.

4 Experimental design

This section will describe the hypotheses behind the research study performed in this paper and will delineate the independent and dependent variables that were studied during both the data collection and analysis phases.

4.1 Hypotheses

The cornerstone of any valid empirical study is the definition of a set of important hypotheses that need to be validated. In our work, the main objective is to devise a set of one or more metrics to evaluate the quality of software product line feature models. More specifically, we are interested in identifying a set of structural measures of feature models that are meaningfully associated with external quality attributes such as *maintainability*. We view maintainability as a complex external quality attribute that according to ISO 9126 (Al-Kilidar et al. 2005) needs to be explained in terms of three subcharacteristics, namely analyzability, changeability, and understandability. We will explain in the subsequent sections that we consider maintainability and its subcharacteristics as the perceived subjective opinion of the model analysts.

The main hypothesis of our work is that structural measures can be developed for feature models that are suitable to serve as indicators of the perceived subjective value of maintainability. This hypothesis is further refined into the following more specific and concrete hypotheses:

H1. A meaningful correlation can be found between software product line feature model structural measures and maintainability;
H2. Feature model structural measures can serve as discriminatory references for maintainability;
H3. Structural measures have different predictive powers for explaining maintainability of feature models.
H4. A relatively small and contained set of structural measures can be used to efficiently evaluate maintainability of any given feature model.

These hypotheses will be evaluated by a set of statistical analysis techniques to clearly understand the relationship between maintainability and the defined structural measures. A correlational study will be used to study whether the defined measures have interdependencies and also to see how well they can be used to explain maintainability (H1). We employ the concept of *entropy* (Fenton and Pfleeger 1997) to study how each of these measures is able to reduce unstructuredness and hence be a discriminatory reference for maintainability (H2). Furthermore, several learning machines and classification algorithms are employed to investigate whether accurate predictive models can be developed to be used as indicators of maintainability as a late external quality attribute from early internal quality attributes (H3). Finally, we will investigate whether a small subset of the proposed metrics can be used to efficiently predict maintainability (H4).

The combination of the results of these studies can provide us with sufficient empirical data to properly evaluate the hypotheses put forth in this section.

4.2 Variables

In order to proceed with the experiments, the defined hypotheses need to be mapped onto a set of measurable independent and dependent variables. These variables are measured in the experiments and will be used in the analysis phase.

4.2.1 Independent variables

The metrics introduced in Table 1 are the independent variables of the study. They are considered as independent since within the *cause–effect* relationship that we are interested in, they represent the *cause*, i.e., we are interested to see whether structural metrics are correlated with maintainability and its subcharacteristics or not. In such a setting, the structural metrics are considered to be the independent variables. Each of these metrics is an independent variable that represents a structural property of a feature model.

4.2.2 Dependent variables

External quality attributes are considered to be the dependent variables. This is due to the fact that they represent the *effect* in the cause–effect relationship as described before. Researchers have argued that since external quality attributes such as maintainability and its subcharacteristics are not easy to measure, it is better to compute the values of structural measures and try to use them as indicators (Kitchenham and Pfleeger 1996). However, since our goal is to validate the hypothesis that structural measures do indeed have a meaningful correlation with maintainability, we need to devise a way to measure the three elements of maintainability.

There have been two main approaches for measuring external quality attributes. The first approach, which we will use in our experiments, is to ask a group of experts to subjectively evaluate the external quality attributes of a set of models. Subsequently, these subjective values are used as measures for the dependent variables (Cruz-Lemus et al. 2009; Genero et al. 2008). The second approach is to define a set of tasks for the experts and then record the time taken for them to complete the tasks (time-to-complete), which would serve as an indication for the dependent variable (Cruz-Lemus et al. 2010; Serrano et al. 2008). We do not use this second approach for three reasons: (1) we are focused on understanding the relation between the subjective perception of model analysts about maintainability and the metric values. Given our focus on the subjective perception of model analysts and that this second approach is essentially an objective measure, it is not appropriate to be used for our purpose; (2) the time-to-complete a given task may be itself dependent on other factors besides the external quality attribute under evaluation. Therefore, such a measurement may not be an accurate indication. In other words, the validation of the fact that "the time-to-complete a given task is a valid measure for an external quality attribute" needs separate independent studies; (3) time-to-complete a given task may be impacted by other factors related to the study subjects such as the preparedness or the attitude of the subjects toward the study being performed during the time of the study, which are irrelevant to the external quality attributes. Such factors may impact the time-to-complete a given task and hence can impact the quality of the drawn conclusions.

In our work, the measurement of the dependent variables was taken using the first approach where the quality of all three subcharacteristics of maintainability were subjectively measured and recorded using questionnaires. This provides us with the required subjective perception of model analysts about maintainability and its subcharacteristics.

Table 4 Metric values for the experiment feature models

Feature model	NF	NTop	NLeaf	CC	CTC	RoV	CoC	FoC	NVC	DT
DELL laptop/notebook computers	46	8	37	110	0.804	5	3.369	0.021	853	3
Inventory	37	2	28	6	0.27	4	1.135	0.162	2028096	4
Sienna	38	4	24	8	0.263	2.642	1.184	0.131	2520	5
HIS	67	7	36	4	0.119	2.129	1.044	0.149	6400	7
Documentation generation	44	3	32	8	0.295	3.583	1.159	0.068	55700000	6
Thread	44	1	26	0	0	2.388	0.977	0.34	80658	13
Smart home	35	6	23	0	0	2.833	0.971	0.571	1048576	3
Arcade game	61	2	47	34	0.557	4.285	1.54	0.081	3.3E+09	8
Model transformation	88	8	55	0	0	2.636	0.988	0.136	1.65E+13	9
Search engine	14	4	10	2	0.285	3.25	1.071	0.357	126	4
Text editor	18	2	10	0	0	2.125	0.944	0.444	396	4
Web Portal	43	5	28	6	0.255	2.8	1.116	0.395	2120800	5
Electronic shopping	287	2	192	21	0.118	3.01	1.069	0.282	2.26E+49	12
Bicycle	27	9	20	2	0.148	3.714	1.037	0.222	1152	5
Mean	76.857	4.571	51.714	9.285	0.195	3.117	1.109	0.274	3.23E+48	6.714
Std. dev.	96.248	2.935	64.24	13.149	0.194	0.714	0.198	0.134	8.54E+48	3.039

4.3 Experimental setup

4.3.1 Objects of study

The feature models that were used in our experiments were all taken from Software Product Line Online Tools (SPLOT) (Mendonca et al. 2009). SPLOT hosts a repository of feature models that are publicly available and distributed by the software product line community. These feature models are stored using the SXFM format,[2] which is an XML serialization of the feature model tree. All of the feature models available in this repository are checked for validity and unreachable dead features prior to being shared with the users. It is important to mention that it contains feature models for different domains of interest and is therefore quite well suited for the purpose of our experimentations.

We selected fourteen feature models from this repository such that the domains were understandable by the participants of our study. The other restriction in the selection of the feature models was that the language of the models be English.

In order to capture the independent variables of the study, we developed an automated tool that would parse the SXFM format of each feature model and would calculate the values of each of the structural metrics. Furthermore, the NVC metric of each feature model was calculated using the binary decision diagram technique proposed by Mendoca et al. (2008). The different feature models used in the experiments along with their metric values are shown in Table 4.

[2] http://fm.gsdlab.org/index.php?title=Format:SXFM.

4.3.2 Data collection

The fundamental purpose of our study was related to identifying any significant relationship between the proposed quality metrics and the subjective perception of domain analysts about the maintainability of a feature model. For this purpose, the values of both the dependent and independent variables need to be collected. The collection of the values for independent variables is quite straightforward and is done automatically by calculating the metric values for each of the feature models. However, the collection of the values for the dependent variables, which are the subjective perception of the participants regarding the three subcharacteristics of maintainability, required input from the human analysts and was therefore performed using questionnaires.

The process that each participant undertook before providing their subjective perception was as follows: each participant was provided with a set of fourteen feature models described in the previous section and was given a period of one week to explore the syntax and semantics of these models. During this period, the participants had the possibility of communicating their questions with the experimenters. After this period, the experimenters held individual assessment meetings with each of the participants during which the subjective opinion of the participant about the subcharacteristics of maintainability was collected using a questionnaire (we will discuss in the threats to validity section of this paper that this setting for gathering information from the participants can pose threats to validity). The questionnaire consisted of 3 sets (for each of the subcharacteristics of maintainability) of 14 questions (for each of the feature models). The questions inquired about the maintainability of the feature model by asking the participants to select one of the seven linguistic values shown in Table 5. In this way, each participant would provide her subjective perception about the maintainability of each feature model using three linguistic values (one for each of the subcharacteristics). The employed linguistic values were originally proposed in (Cruz-Lemus et al. 2009; Genero et al. 2008). As an example, once the participant made up her mind with regard to the understandability of the `Inventory` feature model and decided that it was quite easy to understand it, she would mark the 'quite easy (5)' option on the questionnaire. Each participant would answer such questions with regard to analyzability, understandability, and modifiability of the objects of study.

Some fifteen volunteer subjects participated in the experiments. The participants were all graduate students of Computer Science with the age range between 24 and 29 years. The participants all had a good knowledge of software engineering principles and were given a mini-tutorial on aspects of software product lines and feature models prior to the experimentation week. The subjective opinions of the participants about the subcharacteristics of maintainability for each of the fourteen feature models are given in Table 6. The values reported in this table are the median of the subjective opinions of the participants with regard to each subcharacteristics of maintainability for each feature model, which shows what the participants thought in general about the subcharacteristics of maintainability for each feature model.

Table 5 The linguistic values used for subjectively evaluating the subcharacteristics of maintainability

Extremely difficult	Very difficult	A bit difficult	Neither difficult nor easy	Quite easy	Very easy	Extremely easy
(1)	(2)	(3)	(4)	(5)	(6)	(7)

Table 6 Participants subjective opinion about the maintainability of the experiment feature models according to the linguistic values in Table 5

Feature model	Analyzability	Understandability	Changeability
DELL laptop/notebook computers			
median	2	3	2
Inventory			
median	3	5	4
Sienna			
median	5	4	4
HIS			
median	4	4	4
Documentation generation			
median	2	4	3
Thread			
median	4	5	7
Smart home			
median	3	5	5
Arcade game			
median	2	3	2
Model transformation			
median	2	3	5
Search engine			
median	6	6	5
Text editor			
median	5	5	6
Web portal			
median	3	5	4
Electronic shopping			
median	2	4	4
Bicycle			
median	5	6	5

Now, in order to establish the extent of consensus among the subjective opinions provided by the participants, we perform an *inter-rater reliability* analysis, which shows the degree of agreement between the participants. This is important since if the participants do not reach a relative agreement on their opinions, valid conclusions cannot be drawn from the data. For this purpose, we employ *intra-class correlation (ICC)*. ICC is used to measure the degree of resemblance between the quantitative traits of a group of individuals, i.e., the subjective opinions of the participants with regard to the subcharacteristics of maintainability in our case. Table 7 reports the results of this statistical test based on a two-way random effects model with a confidence interval of 95%.

We have reported both the ICC single measure and the average measure for the subcharacteristics of maintainability. The single measure reliability is used to assess whether the opinion of one participant is apt to be the same as another participant. As seen in Table 7, the single measure reliability of all three subcharacteristics is higher than 0.7, which shows that a reasonable agreement between the participants exists in terms of the values for these subcharacteristics for each of the objects of study. In addition, the average

Table 7 The intra-class correlation (ICC) between the subjective opinions of the participants

Subcharacteristics	ICC single measure	ICC average measure
Analyzability	0.751	0.978
Understandability	0.707	0.973
Modifiability	0.783	0.982

measure reliability provides a means to evaluate how reliable it is to use the mean of the opinions provided by the participants. The ICC average measures for these subcharacteristics as reported in Table 7 are quite high and show high reliability.

5 Analysis techniques

An interesting property of the fourteen feature models that were employed in our experiments is that they are each from a different domain and are therefore a good set of objects for our studies due to their diversity. The feature models are also varied in terms of their metric values as shown by their standard deviation given in Table 4.

The empirical data that were collected are also quantitatively reasonable from the perspective of the amount of data. We obtained 770 data points overall, i.e., 630 subjective opinions from the participants (14 feature models $\times$ 15 participants $\times$ 3 subcharacteristics) and 140 metric values (14 feature models $\times$ 10 structural metrics). The 140 metric values are the values for the independent variables, whereas the 630 subjective opinions are the values for the dependent variables. In the following, given the data that were collected, we explain the different types of statistical analysis processes that were applied on these data in order to better understand the existing relationships and to actually test our hypotheses.

5.1 Correlation study

As the first step of our analysis, we hypothesize that a meaningful correlation between the defined metrics and maintainability can be found (H1). In order to test this hypothesis, we first applied the Kolmogorov–Smirnov test that revealed that our collected data did not have a *normal* distribution. Therefore, a non-parametric test statistic, the Spearman's Rho correlation coefficient (significance level of 0.05) was used to investigate any significance relationships. In this step, three types of correlations were studied:

- *Inter-metric Correlation*: The purpose of this study was to evaluate whether the defined metrics each represent a different aspect of a feature model and are hence unique or in fact some kind of relationship can be found between these metrics;
- *Metrics Indicativeness*: This is the major correlation study of interest, since it investigates whether the defined metrics are useful indicators of maintainability and its subcharacteristics or not;
- *Inter-quality Correlation*: Due to the fact that the external quality attribute under study is comprised of three subcharacteristics, correlation studies are performed between each of these subelements to see whether they themselves are correlated in any way. As one instance, the correlation between *understandability* and *analyzability*, as being constituting elements of maintainability, will be investigated.

Each of these correlation studies reveals some important characteristics of the dependent and independent variables.

5.2 Information gain

The next hypothesis is that the proposed structural metrics are able to serve as discriminatory references for indicating maintainability (H2). The above correlation studies are able to show which of the metrics have a significant relationship with the subelements of maintainability. We further employ the concept of *entropy* (Fenton and Neil 1999) to see how these proposed metrics are able to reduce the uncertainty associated with the correct level of maintainability of a feature model.

Simply stated, entropy is the measure of uncertainty associated with a phenomenon. So in our case, the lack of knowledge about the degree of maintainability of a feature model can be captured in terms of entropy. Now, if the value of one of the proposed metrics is calculated and used for deciding about the maintainability of the model, some useful information about the possible degree of maintainability of that feature model may be gained as a result. For this reason, if a metric has been able to help us reach a better judgement about the maintainability of a feature model, it has been indeed able to reduce our uncertainty and hence reduce the entropy.

The amount of reduced entropy as a result of revealing the correct value for a given metric is known as *information gain* (Kent 1983). It is clear that the higher the information gain for a metric is, the more it is able to reduce the uncertainty about the subcharacteristics of maintainability and can therefore be used as a useful indicator of this external quality attribute. We use the Kullback–Leibler divergence formulation (Hershey and Olsen 2007) for calculating the information gain of each of the proposed metrics. The higher the information gain for a metric is, the more it is able to reveal useful information about the maintainability of a feature model.

5.3 Predictive models

The last two hypotheses (H3 and H4) are related to the predictive power of the defined metrics for maintainability and also the issue that a small set of metrics would be sufficient to accurately evaluate the maintainability of a feature model. To evaluate these hypotheses, we build four learning machines for predicting the three subcharacteristics of maintainability using the proposed metrics. In order to be able to build the learning machines, we create three datasets, one for each of the subcharacteristics of maintainability. Each dataset consists of various rows each of which represents the value of the ten metrics for each of the fourteen feature models, where the elements of each row are the metric values for that feature model and the label to be predicted for that row is the value of one of the subcharacteristics of maintainability.

Given these three datasets, the goal is to build a learning machine that is able to accurately predict the label of a feature model (value of the relevant subcharacteristic of maintainability) based on the values of the structural metrics. In our experiments, we have developed four learning machines to predict the maintainability of a feature model from the *decision tree* and *logistic regression* family of learning machines. These learning machines are commonly used in the domain of software engineering for tasks such as software quality analysis and effort prediction (Khoshgoftaar et al. 2009; Kocaguneli et al. 2009; Liu et al. 2010):

Fig. 4 The summary of the analysis process

- *Decision Trees* are predictive machine learning techniques that identify the value of a dependent variable based on the value of a set of independent variables. Three variations of decision trees, namely *J48*, *ID3*, and *CART*, have been used in our experiments. It is important to mention that the set of metrics used by the decision tree algorithm to build an accurate prediction model will be considered to be the *sufficient subset of the metrics* that are required for predicting the subcharacteristics of maintainability.
- *Logistic Regression* is a method that is employed for predicting the probability of an event by fitting the reference data points into a logistic curve. We have used a *multinomial logistic regression* model in our analysis.

The Waikato Environment for Knowledge Analysis (WEKA) (Garner 1995), which is a widely used suite of machine learning techniques, was employed to learn and test the built learning machines based on our collected independent and dependent variables.

5.4 Summary of analysis

Figure 4 summarizes the process of the analysis performed on the collected data and highlights the purpose of each analysis.

6 Analyses and results

In this section, we describe the results of the analysis performed over the observations made during the experiments. We show the quantitative results of the analysis and discuss how they are relevant for supporting our hypotheses.

6.1 Correlation study

The first technique that we explore is the use of Spearman's Rho correlation in three different contexts, namely to identify relationship between (1) the defined metrics (inter-metric correlation), (2) the metrics and the subcharacteristics of maintainability (metric indicativeness), and (3) the subcharacteristics of maintainability themselves (inter-quality correlation). As usual, a significance level of $\alpha = 0.05$ (95% confidence level) is used to accept the results of the correlation.

6.1.1 Inter-metric correlation

The first study is meant to study the correlation between the proposed metrics (inter-metric correlation), i.e., to see whether some of the metrics capture and cover similar aspects of the feature model and are overlapping or not. The results of this study are shown

in Table 8. In this table, two rows are dedicated to each metric; the values on the upper rows show the degree of correlation, while the ones on the lower rows depict the significance of the correlation. The values in bold represent those that exhibit meaningful correlations and the ones underlined are the most significant correlations. According to Spearman's correlation, a correlation with a significance $p\ value \leq 0.05$ can be considered to be significant, and therefore, in our work, such correlations are considered to be meaningful and are highlighted with bold. Among these highlighted correlation, the one with the higher degree of correlation is regarded as the most significant correlation and is underlined.

Table 8 shows that some of the metrics are in fact correlated and are perhaps overlapping in terms of the concepts that they are representing. For instance, NF and NLeaf are highly correlated, which is an indication that these two metrics are quite similar for a feature model, i.e., a feature model with a high number of features is much likely to have a lot of leaf features. Another example is the relationship between CTC and CoC, which is a sign that a feature model with a higher cross-tree constraint ratio is likely to be more dense and therefore have a larger coefficient of connectivity. On the other hand, a meaningful negative correlation can be seen between CoC and FoC. This means that a feature model which is dense and has a high coefficient of connectivity is relatively less flexible in terms of configuration. In other words, a feature model that is flexible for configuration needs to be less dense.

The correlation found between the metrics in this study shows that all of the proposed metrics are not necessarily required for a comprehensive evaluation of the characteristics of a feature model due to the fact that some of these metrics are highly correlated and can be used interchangeably to some degree, e.g., NF and NLeaf. However, as will be discussed later, the correct subset of the metrics can only be determined based on the characteristic that is being studied, which can be different for dissimilar purposes.

Table 8 Spearman's Rho correlation for the inter-metric study

	NF	DT	NTop	NLeaf	CC	CTC	RoV	CoC	FoC	NVC
NF		**0.68**	0	**0.95**	0.41	0	−0.01	0.24	−0.53	**0.7**
p value		0.01	0.99	0	0.14	1	0.98	0.41	0.05	0.01
DT			−0.32	**0.54**	0.01	−0.27	−0.3	−0.1	−0.2	**0.61**
p value			0.27	0.05	0.96	0.35	0.3	0.73	0.49	0.02
NTop				0.01	−0.06	0.05	0.11	0.01	−0.18	−0.21
p value				0.96	0.83	0.87	0.71	0.98	0.54	0.46
NLeaf					**0.54**	0.17	0.22	0.39	**−0.59**	**0.76**
p value					0.05	0.55	**0.44**	0.17	0.03	0
CC						**0.8**	**0.65**	**0.91**	**−0.67**	0.23
p value						0	0.01	0	0.01	0.42
CTC							0.78	**0.93**	**−0.65**	−0.1
p value								0	0.01	0.73
RoV								**0.69**	−0.47	0.09
p value								0.01	0.09	0.75
CoC									**−0.74**	0.11
p value										0.71
FoC										0.98
p value										0.49

6.1.2 Metrics indicativeness

The second study concentrates on the possibility of a meaningful relationship between the proposed metrics and the three subcharacteristics of maintainability (metric indicativeness study). As it can be seen in Table 9, significant correlations can be found between some of the metrics and the three subelements of maintainability. This shows that the structural metrics defined for a feature can be used as early indicators for external quality attributes that are to be determined late in the development process.

It can be seen from the results of Spearman's Rho correlation that the number of leaves in a feature model has a high negative correlation with both analyzability and understandability. In turn, number of features, which was shown to be highly related to the number of leaves, is also closely correlated with these two characteristics. Therefore, NLeaf and NF can be used as good indicators for analyzability and understandability. Since there is a negative correlation between NLeaf and NF and these two subcharacteristics of maintainability, it can be inferred that the more features and leaf features a feature model has, the more complex it becomes in terms of analyzability and understandability. An interesting and important outcome of this analysis is the corroboration of our earlier assumption that simpler structural metrics may turn out to be useful quality indicators. In our set of metrics, NLeaf and NF are two of the simplest metrics, which appear to be the most useful signs for analyzable and understandable feature models.

Furthermore, changeability is seen to be significantly correlated with the cyclomatic complexity of the feature model. It was shown earlier that CC is itself quite closely related with CoC and CTC, which is also reflected in this analysis. These three metrics, CC, CTC, and CoC, can therefore be considered as good indicators of changeability. An interesting observation that can be made is that just for the purpose of studying the maintainability of a feature model, two metrics would be sufficient, i.e., NLeaf and CC. This is because, from the set of NF and NLeaf that are needed for analyzability and understandability, NLeaf can be selected and also from CC, CTC, and CoC required for changeability, CC can be chosen as a representative. These two metrics (NLeaf and CC) seem to be good indicators of maintainability. The relevance of NLeaf and CC for maintainability can also be logically explained: The two major building blocks of any feature model are its features and integrity constraints. The number of leaves is highly correlated with the number of features and is therefore the representative of the features of a feature model and the cyclomatic complexity, which is calculated by counting the number of integrity constraints. So since these two metrics cover the two main aspects of feature models, it makes sense that they would be highly correlated with the subcharacteristics of maintainability.

Another remarkable observation from this study is that neither NTop nor DT are correlated with any of these three subcharacteristics of maintainability. This shows that these

Table 9 Spearman's Rho correlation for the metric indicativeness study

	NF	DT	NTop	NLeaf	CC	CTC	RoV	CoC	FoC	NVC
Analyzability	**−0.75**	−0.27	0.01	**−0.86**	−0.48	−0.21	−0.39	−0.36	0.49	**−0.77**
p value	**0.00**	0.33	0.96	**0.00**	0.07	0.45	0.15	0.20	0.07	**0.00**
Understandability	**−0.81**	−0.32	−0.07	**−0.82**	**−0.53**	−0.25	−0.12	−0.46	**0.74**	−0.48
p value	**0.00**	0.25	0.79	**0.00**	**0.04**	0.38	0.66	0.09	**0.00**	0.07
Changeability	−0.46	0.04	−0.07	**−0.60**	**−0.92**	**−0.80**	**−0.63**	**−0.89**	**0.73**	−0.32
p value	0.09	0.88	0.78	**0.02**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	0.25

two metrics are not very useful measures to be used in a study which evaluates the maintainability of a feature model.

6.1.3 Inter-quality correlation

The third analysis is concerned with the correlation between the subcharacteristics of maintainability itself. The purpose is to see whether there is some form of conceptual relationship between these quality attributes. As depicted in Table 10 (which follows the same formatting as used in Table 8), all three quality attributes are fairly correlated. Among them, analyzability and understandability seem to have the closest relationship with each other. This was also indicated by the fact that NLeaf metric could be used as the metric to predict both of these quality attributes. The other interesting remark is with regard to the relationship between understandability and the other two quality attributes. Intuitively, it is expected that understandability should be the requirement for the other two quality attributes, which is due to the fact that without understanding a feature model, one cannot proceed with analyzing it or changing it. The fact has also been empirically observed in our analysis where the correlation between understandability and both analyzability (0.74) and changeability (0.63) is slightly higher than the correlation between analyzability and changeability (0.60). The observation can be taken as a sign that understandability is the prerequisite and cornerstone for maintainability and its other two subcharacteristics, meaning that without the proper understanding of the model, analyzability and changeability will not be possible.

6.2 Information gain study

The intention behind using an information gain analysis is to investigate whether the proposed feature model metrics can be used as discriminatory references for maintainability. The metric indicativeness study had a somewhat similar purpose. In this study, we are interested in identifying those metrics that are most helpful in reducing the uncertainty involved in the prediction of maintainability. Assume that the possible space of values for the degree of maintainability of a given feature model is completely unknown because the feature model is still in its early stages of design. The purpose of information gain is to show which one of the structural metrics is able to shed more light on the probable degree of maintainability and hence makes this completely unknown space clearer. In other words, how much would a structural metric help us in reducing our uncertainty toward the perceived values of the subcharacteristics of maintainability for a given feature model. The main difference between the metrics identified in the metric indicativeness study and the information gain analysis is in that although metrics selected in the metric indicativeness study are highly correlated with the subcharacteristics of maintainability, they may not have the discriminatory power required for effectively reducing uncertainty. This is an advantage for the metrics selected by an information gain analysis.

Table 10 Spearman's Rho correlation for the inter-quality study

	Analyzability	Understandability	Changeability
Analyzability		0.74	0.60
p value		0.00	0.02
Understandability			0.63
p value			0.01

Table 11 Information gain study on feature model metrics

Metrics	Analyzability	Understandability	Changeability
NF	0.90	0.87	1.20
DT	0.40	0.37	0.51
Ntop	0.53	0.44	0.58
NLeaf	**1.19**	**1.04**	1.01
CC	0.60	0.71	**1.19**
CTC	0.55	0.95	1.14
RoV	0.60	0.80	0.77
CoC	0.17	0.511	0.76
FoC	**1.10**	**0.95**	**1.35**
NVC	**1.29**	**1.01**	**1.06**

Table 11 shows the results of the information gain study for the three subcharacteristics of maintainability. For each of these characteristics, the top three discriminant metrics are highlighted in bold in the table. We can now compare the results of the information gain analysis with that of the metric indicativeness study. The three metrics selected for analyzability by information gain are NLeaf, FoC, and NVC, while the metrics selected by correlation are NLeaf, NF, and NVC. Interestingly, the two sets of metrics are quite similar where NLeaf and NVC are shared between them. In terms of understandability, the metrics selected by information gain are again NLeaf, FoC, and NVC, and the results of the correlation study are NLeaf, NF, and FoC. There are also two overlaps (NLeaf and FoC) in both of the selected sets of metrics.

It can be inferred that considering both the information gain and correlation studies, NLeaf, NVC, and FoC are most likely to be the best indicators of understandability and analyzability. It was mentioned earlier in the correlation study that NLeaf is a good indicator for these two subcharacteristics. These results reinforce this fact. In addition, NLeaf possesses an advantage over NVC and FoC in that it can be easily calculated by a designer and does not require additional tools to be calculated as is the case for FoC and NVC.

For the case of changeability, CC, FoC, and NVC are chosen by the information gain analysis, whereas CC, CTC, and CoC are selected by the correlation study. Although the two sets only share CC, an obscure observation can be made based on the results shown in Table 8. It can be seen that the metrics in both of these sets are very well correlated with FoC; therefore, FoC can be considered as an important metric for indicating the changeability of a feature model.

As a result of both the information gain and correlation studies, the discriminatory subsets of metrics for analyzability, understandability, and changeability are {NLeaf, NVC, FoC}, {NLeaf, NVC, FoC}, and {CC, FoC, NVC}, respectively. A general conclusion would be to infer that the set of discriminatory metrics for evaluating maintainability of a feature model is {NLeaf, NVC, FoC, CC}. Simply put, from the set of ten metrics proposed in this paper, NLeaf, NVC, FoC, and CC are the most important ones for evaluating maintainability. Furthermore, as corroborated by both of the analyses, DT and NTop are the least important feature model metrics for determining maintainability.

6.3 Predictive models study

The reasons for developing predictive models is to test the last two hypotheses, namely (1) whether predictive models can be built using the structural metrics in order to predict the

maintainability of a feature model (H3) and (2) whether a relatively small subset of the proposed metrics can be chosen in order to efficiently evaluate maintainability (H4). The former of the two hypotheses has already been shown to be valid based on the correlation and information gain studies and will be also further evaluated in this study.

Predictive models are created to best predict the probability of an outcome based on some prior observations. In this study, we will try to build predictive models based on decision trees and logistic regression. These models take the structural metrics of a feature model as input and try to find the most relevant value of maintainability for the given feature model. For building the logistic regression model, we have used the `weka.attributeSelection.CfsSubsetEval` package to eliminate the highly intercorrelated variables (metrics) from the model building process. This variable selection package is based on the work by Hall and Smith (1997).

The exact output that is required from the predictive model is the subjective value for each of the subcharacteristics of maintainability. In other words, given the structural metric values for a feature model, it is expected that the developed predictive models are able to estimate the values for analyzability, understandability, and changeability (a value between 1 and 7).

We employ WEKA to train and test our predictive models. The models that are developed are in the form of J48, ID3, and CART decision trees (Quinlan 1986) and multinomial logistic regression (Hosmer and Lemeshow 2000). For each of the three subcharacteristics of maintainability, one instance of each of the mentioned predictive models is developed (4 model types $\times$ 3 characteristics = 12 predictive models).

In order to evaluate the developed predictive models, we employ two strategies, namely mean absolute error and root mean squared error. These strategies are commonly used in software engineering literature to evaluate the accuracy of the developed techniques (Khoshgoftaar and Seliya 2003; Khoshgoftaar et al. 1992, 1992). Both of the strategies are based on the estimation error of the predictive models. The estimation error is based on the difference between the actual value (ε^a) of the characteristic and the estimated value (ε^e). The first error estimation strategy is *mean absolute error (MAE)*, which is the average of the absolute error of the predictive model:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\varepsilon_i^a - \varepsilon_i^e\right| \tag{1}$$

The second strategy is *root mean squared error (RMSE)*, a quadratic scoring form that estimates the average degree of error of a given predictive model:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(\varepsilon_i^a - \varepsilon_i^e\right)^2}{n}} \tag{2}$$

These two strategies are often used together in the evaluation of a predictive model to diagnose the variation in the errors of the model. A greater difference between the values of MAE and RMSE shows a higher variance in the individual errors. Both of these strategies are negatively oriented in that lower values for MAE and RMSE show a better performance for the predictive model.

Figures 5, 6, and 7 show the degree of error in each of the predictive models. As it can be seen, MAE is in the worst case less than 0.3 out of 7. We can find an upper and lower bound on the accuracy of the predictive models. Since the values of the characteristics to be predicted are natural numbers from 1 to 7, the error of around 0.3 can either be rounded up to 1 for the worst case, or considered as is for the best case. If we consider the worst

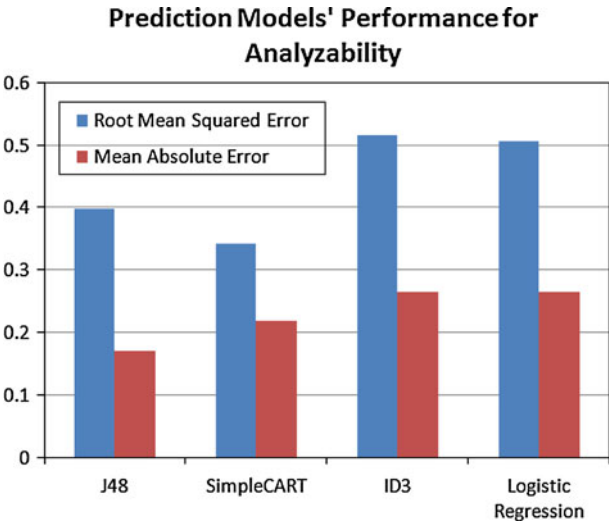

Fig. 5 The accuracy of the predictive models for analyzability

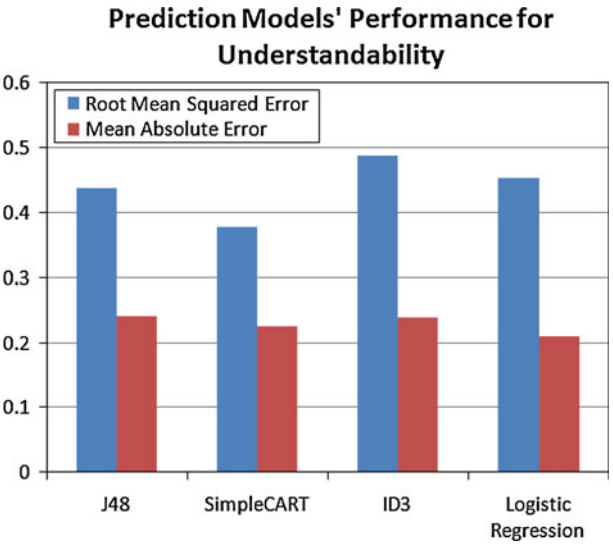

Fig. 6 The accuracy of the predictive models for understandability

case, the accuracy of the predictive models will be $\frac{7-1}{7} = 85\%$; however, for the best case, this is equivalent to $\frac{7-0.3}{7} = 95\%$ accuracy for the predictive model. Even for the worst case, the accuracy rate of the predictive models is quite high and supports our hypothesis that acceptable predictive models can be built from structural metrics in order to predict the subcharacteristics of maintainability.

In addition, it is interesting to view the metrics that have been used in the structure of the developed decision trees. This would help us validate our conclusions on the set of discriminatory metrics for evaluating maintainability of a feature model made based on the

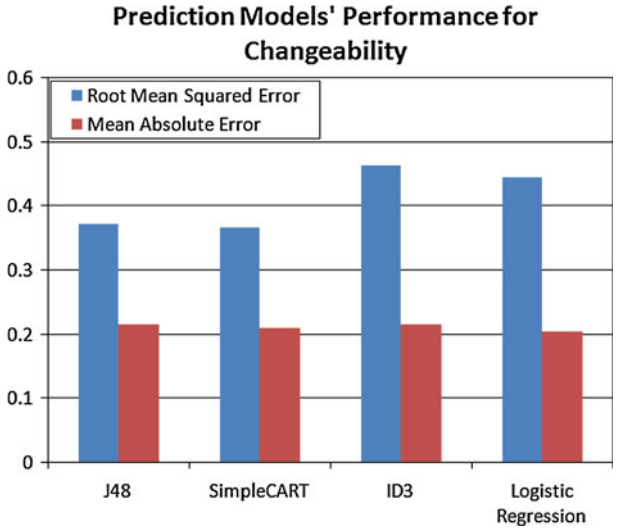

Fig. 7 The accuracy of the predictive models for changeability

correlation and information gain studies. The metrics that have been automatically selected by the decision tree algorithms for predicting analyzability, understandability, and changeability are {NLeaf, NVC}, {NLeaf, FoC}, and {FoC, NLeaf, CC}, respectively. Based on these selected metrics, we can conclude that the set {NLeaf, NVC, CC, FoC} is the sufficient subset of the proposed metrics for evaluating maintainability. It is quite important to mention that this is exactly the same set of metrics that was identified in the information gain and correlation studies. We therefore conclude that these four metrics are the sufficient subset of metrics to be used for predicting the maintainability of a feature model.

7 Discussions

In this section, we provide our analysis on the threats to the validity of our empirical experimentations and will also provide some insight into the lessons learned from these experiments.

7.1 Threats to validity

Empirical evaluation is always subject to different threats that can influence the validity of the results. Here, we will discuss the threats to conclusion, construct, internal, and external validity. We will specifically refer to the aspects of our experiments that may have been affected by these threats.

7.1.1 Conclusion validity

Conclusion validity is the extent to which the conclusions about the presence of a statistically significant relationship between the treatments and the outcomes are valid. In our experiments, a limited number of data points were collected due to our restricted access to appropriate participants that had enough knowledge of software engineering within the

area of software product lines. In addition, only a small collection of publicly available software product line feature models were at our disposal to be used. Among these feature models, many of them were too small to be useful, and therefore, only fourteen feature models could be used in our experiments. Although these numbers are comparable to similar studies in the area of empirical software engineering (Genero et al. 2008; Serrano et al. 2008), they may pose threats to the drawn conclusions. We are currently working on the collection of a larger set of feature models and training of qualified participants to conduct a replication study.

7.1.2 Construct validity

Construct validity is concerned with the extent that the independent and dependent variables provide accurate measurements of what they are intended to measure. The dependent variables: analyzability, understandability, and changeability of a feature model were measured using the subjective opinion of the participants. The threat posed by using subjective measurement mechanisms is that different participants may have different attitudes toward the evaluation of the dependent variables. For instance, some participants may be reluctant to provide high subjective values to the options that they are evaluating, whereas the others may have quite a different mindset. In spite of this fact, this subjective measurement does capture what it claims to measure, which is the degree of each of the subcharacteristics of maintainability for a feature model from the perspective of modeling experts. Our main goal in this paper has been to correlate the perceived subjective value of maintainability with the proposed metrics.

Another important issue that may impact construct validity is the use of a 7-point Likert scale to gather the subjective opinions of the participants. As the Likert scale is ordinal and therefore provides only limited number of options to the participants, it may not provide the participants with the capacity to express their opinions in as precise manner as they would like to. However, despite this threat, empirical research has shown that the best number of options for Likert scale is between 4 and 7 (Lozano and García-Cueto 2008). This is because although more than 7 options will give better *psychometric properties*, they are in many cases likely to exceed the *discriminative capacity* of the participants. There are also additional threats to construct validity as a result of using the Likert scale, which is due to the fact that Likert scale is an ordinal scale. According to Hubbard and Evans (2010), there are three important threats caused by ordinal scales: (1) the ordinal labels employed in ordinal scales such as the Likert scale could be inconsistently interpreted by different users or even by the same user under different circumstances; (2) many users treat ordinal scales as if they had the properties of ratio scales and hence could provide unreliable information; and (3) the distance between the different labels of an ordinal scale is not deterministically known, and therefore, clear comparison between the significance of various ordinal labels could be hard for an ordinary user to do. For these reasons, it is important to notice that the use of the ordinal Likert scale could have impacted the construct validity of our experimentations.

7.1.3 Internal validity

Internal validity is the degree that conclusions can be made about the causal effect of independent variables on dependent variables. An internally invalid experiment can lead to results that are not necessarily inferred from a causal relationship. For this reason, we investigate the following issues:

- *Difference between subjects:* In the study reported in this paper, there was no significant difference between the experience of the participants in using software product line feature models, due to the fact that all of them were Computer Science graduate students that had taken at least one advanced software engineering course. Therefore, error variance from difference between subjects is kept to a possible minimum.
- *Knowledge of the universe of discourse:* Each of the feature models represented different domains of interest. However, the models were simple enough to be understandable by the participants. In addition, the participants were given enough time (1 week) and support to become familiar with the concepts in each feature model. Hence, knowledge of the domain was not a significant impacting factor in the study.
- *Maturation effects:* The experiments were performed in a limited time frame of one week during which the participants did not become involved in any other training exercises in the area of software engineering to minimize the maturation/learning effect.
- *Fatigue effects:* The sessions held by the experimenters with each individual participant to collect the subjective opinions were limited to 1 hour. Since the participants are all Computer Science students and a usual lecture time is 90 min, the fatigue effect is not significant in this study.
- *Persistence effects:* The study was performed on subjects that had no prior experience in participating in similar studies. Therefore, persistence effects were avoided.
- *Subject motivation:* The participants were all graduate students and were therefore quite aware of the importance and impact of empirical evaluations for our research. Furthermore, the experiments were done on a volunteer basis, and it is fair to assume that the participants were quite motivated to take part in the study.

Plagiarism and influence were controlled by explicitly asking the subjects not share any information about the study with each other until after the experiments were concluded. Furthermore, the evaluation sessions were held individually for each participant with the presence of an experimenter. In addition, the use of the 7-point Likert scale in our experiments could have impacted the internal validity of the experiments due to the discrete nature of this ordinal scale in capturing the participants' input.

The other operational aspect of our experiments that could impact validity is related to how the subjective values were gathered from the participants: In order to remove the rating bias in the subjective opinions of the participants and also to help the participants in the process, one of the experimenters was involved with the participants while they were asserting their subjective opinions. This could have a potential impact on the opinion of the participant and effect validity. Although given the fact that the experimenters were unbiased toward the subjective opinions of the participants, still they could have influenced the participants. We will consider removing this threat to validity in our future experiments.

7.1.4 External validity

External validity is the extent that the obtained results of a study can be generalized to the setting under study and other relevant research scenarios. The results of a completely externally valid study can be generalized and applied safely to the software engineering practice and be recommended as blueprints. The following two issues were considered for external validity:

- *Materials and tasks used:* In our experiments, we tried to use the best quality feature models that are publicly available to the research community. These feature models

cover a wide range of different domains and are representatives of real-world models. However, further empirical studies need to be conducted on more complex feature models.

- *Subjects:* Due to the difficulty of gathering professional opinions about the subcharacteristics of maintainability, we used Computer Science graduate students in our studies. In our experiments, we do not particularly need a high level of industrial experience to be able to complete the experiment. Furthermore, authors such as Basili et al. (1999) believe that in many cases, the use of students in experiments has little impact on the validity of the results. However, further experiments involving industrial professional subjects are needed to ensure the external validity of our experiments.

Another threat to validity may be related to the tool that was used to work with the feature models. In our experiments, we used the online feature model editor provided by SPLOT (Mendonca et al. 2009). It is possible that the results of the experiments were affected by the usability of this tool that was used for the manipulation of the feature models; however, since the tool was used by all of the participants, we believe that it possibly affected all of the participants in the same way. Unfortunately, the effect of this threat cannot be controlled by our experiments.

7.2 Lessons learned

The process of the design and execution of the research work required for this paper provided us with important insights into various aspects of performing empirical investigation for the field of software product lines.

The main challenge that we faced was with regard to access to a set of well-established and publicly available software product line feature models. Fortunately, researchers at the University of Waterloo have set up a public repository (SPLOT) where feature models can be shared. However, to date, only a limited number of feature models have been shared which are mostly academic feature models that have been designed for research purposes. It is important to gather industrial feature models inside such repositories to strengthen future empirical studies and evaluations. Our future steps will involve spending effort on collecting industrial scale feature models.

Another issue was the need for objective mechanisms for measuring the value of the dependent variable, i.e., the subcharacteristics of maintainability. Several researchers have opted for using the time-to-complete the required tasks as an indication of the dependent variable. This assumption has not yet been empirically validated. Moreover, it seems that the time required for completing a task is dependent on many contextual factors that might affect the construct validity of the study. Our experience shows that although subjective opinions of the participants are good measures of the dependent variable, still the use of objective measures to second the obtained results is valuable. We believe that both theoretical and empirical development of appropriate objective measures for the field of software product line feature models is required.

The third important lesson learnt was with regard to the complexity of the structural metrics. Our intuition and the results of prior experimentations on software metrics supported the idea that simple metrics perform quite well for indicating external quality attributes. The results of the current study corroborated this fact in that simple structural metrics are in fact quite useful. It is important to mention that more complex metrics do not necessarily result in better software metrics, at least for the area of software product line feature models.

An important additional observation with respect to the nature of the structural metrics was also made. As was discussed earlier, the set {NLeaf, CC, NVC, FoC} was identified to be the sufficient subset of the proposed metrics for evaluating maintainability. These four metrics can be viewed in terms of the major characteristics of a feature model, i.e., the number of features and integrity constraints in a feature model along with the proportion of optional features within the feature model and the number of possible configurations of the feature model are the major influences on maintainability. This shows that as the number of features and integrity constraints in a feature model increases and as a result the number of possible configurations grows, the maintenance of the feature model becomes harder; however, at the same time, the existence of more optional features in the feature model can alleviate this negative impact and hence enhance maintainability. Therefore, NLeaf, NVC, and CC on the one hand (negative impact on maintainability) and FoC on the other hand (positive influence on maintainability) form a balance over the maintainability of a feature model. The right balance between these metrics should be found in order to have a maintainable feature model.

Overall, the results of the study show that we can reasonably claim that our hypotheses have been confirmed. The main implication of this is that structural metrics can indeed be used as early indicators of maintainability as an external quality attribute of software product line feature models. However, further experimentation with industrial scale feature models and more participants are required to fully verify the conclusions of our work.

8 Related work

The major direction of research in software product line feature models has been toward the development and validation of product configurations based on a given set of stakeholder hard constraints (Janota and Kiniry 2007; Benavides et al. 2005; Batory 2005; Boskovic et al. 2010; Bagheri et al. 2010). Feature model configurations have been often validated and verified using Logic Truth Maintenance Systems (LTMS). Three of the most widely used methods in this context are constraint satisfaction problem (CSP) solvers (Benavides et al. 2007), propositional SATisfiability problem (SAT) solvers (Batory 2005), and the binary decision diagrams (BDD) (Mendonca et al. 2008). Furthermore, Description Logic reasoners have also been employed to validate the structural correctness of feature model configurations (Wang et al. 2007).

However, very limited research has been conducted in the area of software product line quality engineering tasks such as defining quality metrics, evaluating external and internal quality attributes and building predictive models for quality prediction and maintenance. More specifically, only a few researchers have addressed the issue of developing appropriate structural quality metrics for software product line feature models. From among these few sets of quality metrics that have been proposed by the researchers (de Oliveira et al. 2008; Zhang et al. 2008), none of them has been considered and defined within the premise of measurement theory. Therefore, the metrics that have been defined seem to be rather ad hoc in that they have not been analyzed in any way to show whether they satisfy any of the important properties as required by a valid metric definition like the DISTANCE framework nor have they been analyzed to show whether they have any interesting properties, e.g., properties proposed by other authors such as Briand et al. (1996) and Weyuker (1988). More importantly, the metrics in the related work have not been analyzed with respect to their relationship with external quality attributes and have just been proposed for the sake of measurement. Although such an approach to the definition of software metrics creates a set of feature model metrics, the metrics might not necessarily be

useful measures for evaluating the properties of the feature model, e.g., the external quality attributes of the model.

To the extent of our knowledge, SDMetrics (de Oliveira et al. 2008) is the only available measurement suite for feature models that provides a set of metrics based on the correspondence that it creates between feature models and UML Class diagrams. It defines four metrics based on object-oriented design measures to evaluate feature models, which are essentially based on the Weighted Methods per Class (WMC) metric. In another attempt (Zhang et al. 2008), formal vADL specifications have been used to specify and measure the quality of software product line architectures. This work has a rather different flavor in that it considers the quality of software product line architectures rather than our focus which is on software product line feature models. In their work, the authors focus on architectural descriptions provided in vADL. Since vADL is based on pi-calculus, its properties have been used to define a set of metrics called PLA metrics such as similarity, variability, reusability, and complexity. Both of these two works lack the required theoretical rigor for their evaluation. Furthermore, they have not been empirically studied to see whether any meaningful correlation can be found between these metrics and external quality attributes.

The other work in software product line quality engineering is rather far from the focus of our work. Some researchers have made a correspondence between the need for quality engineering in software product lines and quality assurance in software architectures. For instance, the Holistic Product Line Architecture Assessment (HoPLAA) is an adaptation of the Architecture Tradeoff Analysis Method (ATAM) (Kazman et al. 1998) for the area of software product lines (Olumofin and Mišić 2007). As another example, the authors of Etxeberria and Sagardui (2008) have extended FeatuRSEB (Griss et al. 1998) with the quality attribute utility tree introduced in ATAM (Kazman et al. 1998). Similarly, the FODA feature modeling formalism (Kang et al. 1990) has been combined with the *i** goal-oriented modeling framework (Yu and Mylopoulos 1994) to form the F-SIG framework to address quality attributes in feature models (Jarzabek et al. 2006). These works only focus on architectural aspects of software product lines. This further highlights the need for the kind of theoretical and empirical evaluation that we have conducted in order to define appropriate software product line feature model structural metrics and employ them as suitable early indicators of external quality attributes. We believe that our proposed work can be seen as a first step toward the much needed empirical research on the relation between feature model structural metrics and external quality attributes. Here, we have only studied this relation for maintainability and its subcharacteristics.

9 Concluding remarks

Software product line engineering is a paradigm for covering re-use-based software engineering through capturing the commonalities and variabilities between the applications of a target domain. Feature models are one of the main means for facilitating this process. Since feature models serve as a platform for deriving many applications from a software product line, their quality influences the final properties of the developed applications. Therefore, it is important to consider ensuring the quality of feature models from the early stages of their development. To serve this purpose, our current work builds on the hypothesis that the structural measures of a model influence its cognitive complexity, which will in turn have impact on the its external quality attributes.

For this reason, we have proposed and both theoretically and empirically validated a set of structural feature model metrics. We have analyzed the suitability of these metrics for

the purpose of serving as early indicators of maintainability. Based on our empirical study, a sufficient subset of metrics have been identified that are directly correlated with maintainability and its subcharacteristics, which can be hence used as suitable indicators.

We are currently working on improving the complexity of the feature models used in our study along with training more participants in order to perform further replication studies. Like any other empirical study, further experimentations are required to validate our results and draw final conclusions.

The other direction of future work that we are interested in studying is the empirical exploration of the possible corrective actions that experienced feature model designers take once they encounter a feature model with poor maintainability. As a part of our work, we will provide a group of feature model designers with a set of poor maintainability feature models and ask the designers to take corrective steps, e.g., refactor the feature model, remove some redundant features or constraints, among others. These steps will be observed, recorded, and analyzed in order to find the best actions for different situations for improving the maintainability of a feature model. Given our current work that provides the basis for identifying feature models with poor maintainability, it is possible to identify the objects of analysis for this line of future work and perform further empirical experiments. An alternative to this approach would be to analyze the change history of the feature models as is done in related areas (Herrmannsdoerfer et al. 2010). Unfortunately, this can hardly be done due to lack of public repositories of feature models with a complete revision history.

In both of these lines of future work, we will consider removing the threats to validity that were discussed in this paper as much as possible. Specifically, we are redesigning the experimental setting to minimize the influence of the experimenters on the subjective opinion of the participants.

Acknowledgments The authors would like to acknowledge the many valuable suggestions made by the anonymous reviewers of the paper.

References

Al-Kilidar, H., Cox, K., & Kitchenham, B. (2005). The use and usefulness of the ISO/IEC 9126 quality standard. In *2005 International Symposium on Empirical Software Engineering, 2005* (p. 7). IEEE.

Babar, M., Chen, L., & Shull, F. (2010). Managing variability in software product lines. *IEEE Software, 27*(3), 89–91.

Bagheri, E., Noia, T. D., Ragone, A., & Gasevic, D. (2010). Configuring software product line feature models based on stakeholders' soft and hard requirements. In *The 14th International Software Product Line Conference*. Springer.

Bansiya, J., & Davis, C. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering, 28*(1), 4–17.

Barbacci, M., Klein, M., Longstaff, T., & Weinstock, C. (1995). Quality attributes. *SEI*, December.

Basili, V., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering, 25*(4), 456–473.

Batory, D. (2005). Feature models, grammars, and propositional formulas. *Lecture Notes in Computer Science, 3714*, 7.

Benavides, D., Segura, S., Trinidad, P., & Ruiz-Cortes, A. (2007). FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the 1st International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)* (pp. 129–134).

Benavides, D., Trinidad, P., & Ruiz-Cortes, A. (2005). Automated reasoning on feature models. In *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005* (Vol. 3520, pp. 491–503). Springer.

Boskovic, M., Bagheri, E., Gasevic, D., Mohabbati, B., Kaviani, N., & Hatala, M. (2010). Automated staged configuration with semantic web technologies. In *International Journal of Software Engineering and Knowledge Engineering*. World Scientific.

Briand, L., Bunse, C., & Daly, J. (2001). A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. *IEEE Transactions on Software Engineering, 27*(6), 513–530.

Briand, L., Morasca, S., & Basili, V. (1996). Property-based software engineering measurement. *IEEE Transactions on Software Engineering, 22*(1), 68–86.

Briand, L., Wust, J., Daly, J., & VictorPorter, D. (2000). Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software, 51*(3), 245–273.

Briand, L. C., Wüst, J., Ikonomovski, S. V., & Lounis, H. (1999). Investigating quality factors in object-oriented designs: An industrial case study. In *ICSE'99: Proceedings of the 21st International Conference on Software engineering* (pp. 345–354). New York, NY: ACM.

Cant, S., Henderson-Sellers, B., & Jeffery, D. (1994). Application of cognitive complexity metrics to object-oriented programs. *Journal of Object Oriented Programming, 7*, 52–52.

Chidamber, S., Kemerer, C., & MIT, C. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering, 20*(6), 476–493.

Cruz-Lemus, J. A., Genero, M., Manso, M. E., Morasca, S., & Piattini, M. (2009). Assessing the understandability of uml statechart diagrams with composite states—A family of empirical studies. *Empirical Software Engineering, 14*(6), 685–719.

Cruz-Lemus, J. A., Maes, A., Genero, M., Poels, G., & Piattini, M. (2010). The impact of structural complexity on the understandability of uml statechart diagrams. *Information Science, 180*(11), 2209–2220.

Czarnecki, K., Helsen, S., & Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice, 10*(1), 7–29.

de Oliveira, E. A. Jr., Gimenes, I. M. S., & Maldonado, J. C. (2008). A metric suite to support software product line architectures. In *XXXIV Conferncia Latinoamericana de Informtica (CLEI 2008)*.

Etxeberria, L., & Sagardui, G. (2008). Variability driven quality evaluation in software product lines. In *Proceedings of the 12th International Software Product Line Conference* (pp. 243–252). IEEE.

Fenton, N. (1994). Software measurement: A necessary scientific basis. *IEEE Transactions on software engineering, 20*(3), 199–206.

Fenton, N., & Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on Software Engineering, 25*(5), 675–689.

Fenton, N., & Neil, M. (1999). Software metrics: Successes, failures and new directions. *Journal of Systems and Software, 47*(2–3), 149–157.

Fenton, N., & Pfleeger, S. L. (1997). *Software metrics: A rigorous and practical approach* (2nd ed.). Boston, MA: PWS Publishing Co.

Finkelstein, L. (2003). Widely, strongly and weakly defined measurement. *Measurement, 34*(1), 39–48.

Garner, S. (1995). Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand Computer Science Research Students Conference* (pp. 57–64). Citeseer.

Genero, M., Olivas, J., Piattini, M., & Romero, F. (2001). Using metrics to predict OO information systems maintainability. In *Advanced Information Systems Engineering* (pp. 388–401). Springer.

Genero, M., Poels, G., & Piattini, M. (2008). Defining and validating metrics for assessing the understandability of entity-relationship diagrams. *Data and Knowledge Engineering, 64*(3), 534–557.

Griss, M., Favaro, J., & dAlessandro, M. (1998). Integrating feature modeling with the RSEB. In *Proceedings of the 5th International Conference on Software Reuse* (pp. 76–85). Citeseer.

Hall, M., & Smith, L. (1997). Feature subset selection: A correlation based filter approach. In *Proceedings of the 4th International Conference on Neural Information Processing Systems* (pp. 855–858).

Herrmannsdoerfer, M., Ratiu, D., & Koegel, M. (2010). Metamodel usage analysis for identifying metamodel improvements. In *Proceedings of the 3rd International Conference on Software Language Engineering*.

Hershey, J., & Olsen, P. (2007). Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2007*, Vol. 4, ICASSP 2007.

Hitz, M., & Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems. *Proceedings of the International Symposium on Applied Corporate Computing, 50*, 75–76.

Hosmer, D., & Lemeshow, S. (2000). *Applied logistic regression*. New York: Wiley.

Hubbard, D., & Evans, D. (2010). Problems with scoring methods and ordinal scales in risk assessment. *IBM Journal of Research and Development, 54*(3), 2–10.

Janota, M., & Kiniry, J. (2007). Reasoning about feature models in higher-order logic. In *Software Product Line Conference, 2007. SPLC 2007. 11th International* (pp. 13–22).

Jarzabek, S., Yang, B., & Yoeun, S. (2006). Addressing quality attributes in domain analysis for product lines. *IEE Proceedings-Software, 153*, 61.

Jones, C. (2008). *Applied software measurement*. New York: McGraw-Hill Osborne Media.

Kang, K., Cohen, S., Hess, J., Novak, W., & Peterson, A., C.-M. U. P. P. S. E. INST. (1990). *Feature-oriented domain analysis (FODA) feasibility study*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The architecture tradeoff analysis method. In *iceccs*. Published by the IEEE Computer Society, p. 0068.

Kent, J. (1983). Information gain and a general measure of correlation. *Biometrika, 70*(1), 163.

Khoshgoftaar, T. M., & Seliya, N. (2003). Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering, 8*(3), 255–283.

Khoshgoftaar, T. M., Munson, J. C., Bhattacharya, B. B., & Richardson, G. D. (1992). Predictive modeling techniques of software quality from software measures. *IEEE Transactions on Software Engineering, 18*(11), 979–987.

Khoshgoftaar, T. M., Rebours, P., & Seliya, N. (2009). Software quality analysis by combining multiple projects and learners. *Software Quality Journal, 17*(1), 25–49.

Khoshgoftaar, T., Bhattacharyya, B., & Richardson, G. (1992). Predicting software errors, during development, using nonlinear regression models: A comparative study. *IEEE Transactions on Reliability, 41*(3), 390–395.

Kitchenham, B., & Pfleeger, S. (1996). Software quality: The elusive target. *IEEE Software, 13*(1), 12–21.

Kleppe, A. G., Warmer, J., & Bast, W. (2003). *MDA explained: The model driven architecture: Practice and promise*. Boston, MA: Addison-Wesley Longman Publishing Co Inc.

Kocaguneli, E., Kultur, Y., & Bener, A. (2009). *Combining Multiple Learners Induced on Multiple Datasets for Software Effort Prediction*. ISSRE.

Korson, T., & McGregor, J. D. (1990). Understanding object-oriented: A unifying paradigm. *Communications of the ACM, 33*(9), 40–60.

Lee, K., Kang, K., & Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering. *Lecture Notes in Computer Science, 2319*, 62–77.

Liu, Y., Khoshgoftaar, T. M., & Seliya, N. (2010). Evolutionary optimization of software quality modeling with multiple repositories. In *IEEE Transactions on Software Engineering, 99*, no. PrePrints.

Lopez-Herrejon, R., Batory, D. (2001). A standard problem for evaluating product-line methodologies. In *Lecture Notes in Computer Science*, pp. 10–24.

Lozano, L. M., García-Cueto E., & Muñiz, J. (2008). Effect of the number of response categories on the reliability and validity of rating scales. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, 4*(2), 73–79. [Online]. Available: http://dx.doi.org/10.1027/1614-2241.4.2.73.

Manso, M., Genero, M., & Piattini, M. (2010). No-redundant metrics for UML class diagram structural complexity. In *Advanced Information Systems Engineering* (pp. 1029–1029). Springer.

McGregor, J. D., Muthig, D., Yoshimura, K., & Jensen, P. (2010). Guest editors' introduction: Successful software product line practices. *IEEE Software, 27*(3), 16–21.

Mendonca, M., Branco, M., & Cowan, D. (2009). S.p.l.o.t.: Software product lines online tools. In *OOPSLA'09: Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications* (pp. 761–762). New York, NY: ACM.

Mendonca, M., Wasowski, A., Czarnecki, K., & Cowan, D. (2008). Efficient compilation techniques for large scale feature models. In *Proceedings of the 7th International Conference on Generative Programming and Component Engineering* (pp. 13–22). New York, NY: ACM.

Olumofin, F. G., & Mišić, V. B. (2007). A holistic architecture assessment method for software product lines. *Information and Software Technology, 49*(4), 309–323.

Poels, G., & Dedene, G. (1999). DISTANCE: A framework for software measure construction. *Technical Report, KU Leuven-Departement toegepaste economische wetenschappen*.

Poels, G., & Dedene, G. (2000). Distance-based software measurement: Necessary and sufficient properties for software measures. *Information and Software Technology, 42*(1), 35–46.

Pohl, K., & Metzger, A. (2006). Software product line testing. *Communications of the ACM, 49*(12), 78–81.

Pohl, K., Böckle, G., & Van DerLinden, F. (2005). *Software product line engineering: Foundations, principles, and techniques*. Berlin, Heidelberg, New York: Springer.

Quinlan, J. (1986). Induction of decision trees. *Machine learning, 1*(1), 81–106.

Scott, W. (1998). *Organizations: Rational, natural, and open systems*. Upper Saddle River, NJ: Prentice hall.

Selic, B. (2006). Model-driven development: Its essence and opportunities. In *ISORC'06: Proceedings of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing* (pp. 313–319). Washington, DC: IEEE Computer Society.

Serrano, M. A., Calero, C., Sahraoui, H. A., & Piattini, M. (2008). Empirical studies to assess the understandability of data warehouse schemas using structural metrics. *Software Quality Journal, 16*(1), 79–106.

Siau, K., & Tan, X. (2005). Improving the quality of conceptual modeling using cognitive mapping techniques. *Data and Knowledge Engineering, 55*(3), 343–365.
Simon, H. (1978). Rationality as process and as product of thought. *The American Economic Review, 68*(2), 1–16.
Tessier, P., Gérard, S., Terrier, F., & Geib, J. (2005). "Using variation propagation for model-driven management of a system family." In *Software Product Lines*, pp. 222–233.
Wang, H., Li, Y., Sun, J., Zhang, H., & Pan, J. (2007). Verifying feature models using OWL. *Web Semantics: Science, Services and Agents on the World Wide Web, 5*(2), 117–129.
Weiss, D. M., Clements, P. C., Kang, K., & Krueger, C. (2006). Software product line hall of fame. In *SPLC'06: Proceedings of the 10th International on Software Product Line Conference* (p. 237). Washington, DC: IEEE Computer Society.
Weyuker, E. (1988). Evaluating software complexity measures. *IEEE Transactions on Software Engineering, 14*, 1357–1365.
Yu, E., & Mylopoulos, J. (1994). Understanding "why" in software process modelling, analysis, and design. In *Proceedings of the 16th international conference on Software engineering* (p. 168). IEEE Computer Society Press.
Zhang, T., Deng, L., Wu, J., Zhou, Q., & Ma, C. (2008). Some metrics for accessing quality of product line architecture. In *CSSE'08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering* (pp. 500–503). Washington, DC: IEEE Computer Society.

Author Biographies

Ebrahim Bagheri is currently an assistant professor at the AU School of Computing and Information Systems, and is affiliated with the Institute for Information Technology at the National Research Council Canada as a visiting researcher. He completed his PhD degree at the Faculty of Computer Science, University of New Brunswick, and specializes in topics ranging from the (meta)modeling of complex interconnected systems to collaborative information systems design. Currently, his research focuses on two areas namely, quality engineering for software product lines; and social informatics for healthcare. His work on collaborative modeling is one of a kind in providing tools and techniques for collaborative risk management and quality engineering. He has extensively published over 60 papers in top tier international journals and conferences including the IEEE Transaction on Systems, Man, and Cybernetics Part A, Elsevier journal of Systems and Software, Springer Information Systems Frontiers Journal, Springer Knowledge and Information Systems Journal and others. He can be reached http://glass.cs.unb.ca/∼ebrahim.

Dragan Gaševic is a Canada Research Chair in Semantic Technologies and an Associate Professor in the School of Computing and Information Systems at Athabasca University. He is also an Adjunct Professor in the School of Interactive Arts and Technology at Simon Fraser University and an associated research member of the GOOD OLD AI Research Network at the University of Belgrade. He is a recipient of Alberta Ingenuity's 2008 New Faculty Award. His research interests include semantic technologies, software language engineering, technology-enhanced learning, and service-oriented architectures. He has (co-)authored more than 200 research papers. He has been serving on editorial boards of three international journals and has edited special issues in journals such as IET Software and IEEE TSE. He has been the organizer, chair, and member of program committees of many international conferences.