Journal of Universal Computer Science, vol. 19, no. 1 (2013), 25-52
submitted: 1/1/12, accepted: 28/12/12, appeared: 1/1/13 © J.UCS

Systematic Evaluation of Software Product Line Architectures

Edson A. Oliveira Junior and Itana M. S. Gimenes
(State University of Maringá, DIN-UEM, Maringá-PR, Brazil
{edson,itana}@din.uem.br)

José C. Maldonado and Paulo C. Masiero
(University of São Paulo, ICMC-USP, São Carlos-SP, Brazil
{jcmaldon,masiero}@icmc.usp.br)

Leonor Barroca
(The Open University, Milton Keynes, United Kingdom
l.barroca@open.ac.uk)

Abstract: The architecture of a software product line is one of its most important artifacts as it represents an abstraction of the products that can be generated. It is crucial to evaluate the quality attributes of a product line architecture in order to: increase the productivity of the product line process and the quality of the products; provide a means to understand the potential behavior of the products and, consequently, decrease their time to market; and, improve the handling of the product line variability. The evaluation of product line architecture can serve as a basis to analyze the managerial and economical values of a product line for software managers and architects. Most of the current research on the evaluation of product line architecture does not take into account metrics directly obtained from UML models and their variabilities; the metrics used instead are difficult to be applied in general and to be used for quantitative analysis. This paper presents a Systematic Evaluation Method for UML-based Software Product Line Architecture, the SystEM-PLA. SystEM-PLA differs from current research as it provides stakeholders with a means to: (i) estimate and analyze potential products; (ii) use predefined basic UML-based metrics to compose quality attribute metrics; (iii) perform feasibility and trade-off analysis of a product line architecture with respect to its quality attributes; and, (iv) make the evaluation of product line architecture more flexible. An example using the SEI's Arcade Game Maker (AGM) product line is presented as a proof of concept, illustrating SystEM-PLA activities. Metrics for complexity and extensibility quality attributes are defined and used to perform a trade-off analysis.

Key Words: Quality attributes, metrics, product line architecture evaluation, trade-off analysis, UML, variability

Category: D.2 (Software Engineering), D.2.11 (Software Architectures)

1 Introduction

A Software Product Line (PL) [Pohl et al. 05, Linden et al. 07] represents a set of systems sharing common features that satisfy the needs of a particular market or mission segment. This set of systems is also called a product family. The family's members are specific products developed in a systematic way from the

PL core assets. The core assets have a set of common features as well as a set of variable parts, which represent later design decisions [Pohl et al. 05]. The composition and the configuration of such assets yield specific products.

The PL architecture (PLA) plays a central role to successfully generate specific products taking into account the development and evolution of a PL. It abstractly represents the architecture of all potential PL products from a specific domain. The PLA addresses the PL design decisions by means of their similarities, as well as their variabilities [Taylor et al. 09]. Organizations should continuously evaluate the quality of their products by managing their PL evolution and variabilities. Thus, the PLA evaluation should be taken into consideration as one of the most important activities throughout a PL life cycle [Linden et al. 07].

Architecture evaluation is an important activity of software design. Informal evaluations, based on use case scenarios, for instance, are widely performed. However, most of the time they do not generate accurate results [Linden et al. 07]. Although there are more rigorous and consolidated evaluation methods in the literature, such as Architecture Tradeoff Analysis Method (ATAM) and Software Architecture Analysis Method (SAAM) [Clements et al. 02], the evaluation of a PLA [Pohl et al. 05, Linden et al. 07] requires particular attention due to variability issues. Such an evaluation should take into account issues such as: the relevant quality attributes of a PLA; the time when the PLA is evaluated; and the techniques and metrics used to evaluate the PLA [Etxeberria and Sagardui 08].

The evaluation of a quality attribute-based PLA can be used as a parameter for evaluating a PL in general [Etxeberria and Sagardui 08]. By trading-off the PLA quality attributes it is possible for PL managers and architects to prioritize which quality attribute must be taken into consideration during PLA evolutions. This occurs because the PLA quality attributes take into account variabilities, which can be used as a parameter to the quality evaluation of an overall PL. The evaluation of a PLA also requires a set of basic and quality attribute metrics that can provide evidence of the PL quality, thus serving as the basis to analyze the managerial and economical values of a PL [Böckle et al. 04].

This paper presents SystEM-PLA, a **Syst**ematic **E**valuation **M**ethod for UML-based **PLA**. SystEM-PLA allows both PLA quality attribute and structural evaluations based on, respectively, scenarios and metrics. SystEM-PLA provides a means to perform quantitative and qualitative analysis. The main contributions of this paper are: (i) provide product line architects with a means to evaluate an architeture based on metrics and trade-off analysis to prioritize quality attributes; (ii) allow product line managers to analyze a product line by means of its produced products and, therefore, take such an analysis as a tool for return on investment and product line evolution; and (iii) establish a practical method to perform quantitative and qualitative analysis of product line architetures, by combining best practices of consolidated techniques, such as, ATAM and GQM and incorporating product line variability issues partially tackled by EATAM and HoPLSAA.

This paper is organized as follows: Section 2 analyzes related work; Section 3 introduces SystEM-PLA; and, Section 4 presents the conclusion and directions for future work.

2 Related Work

SystEM-PLA differs from current approaches in the literature, in that it provides a means to: (i) estimate and analyze, quantitatively and qualitatively, the potential PL products that can be generated from a PLA based on its models and variabilities by correlating quality attributes to variabilities and defining variability-based scenarios to support trade-off analysis; (ii) allow stakeholders to instantiate an evaluation meta-process based on guidelines in order to properly guide the execution of PLA evaluations; (iii) allow the use of predefined basic metrics and the composition of new metrics for PLA quality attributes; (iv) apply statistics and combinatorics in order to analyze the feasibility of a PLA with respect to its quality attributes for a certain domain; and (v) make PLA evaluations more flexible in order to apply them to different contexts.

[Barbacci et al. 10], [Etxeberria and Sagardui 08] and [Gannod and Lutz 00] as well as [Dolan et al. 00] and [Riva and Rosso 03] propose approaches for evaluating PLA quality attributes. [Gannod and Lutz 00] propose an evaluation process in which the relationship between quality attributes and variabilities is described at the ADL level. [Riva and Rosso 03] relate scenarios to quality attributes to identify architectural problems, such as, architecture evolution and reconstruction, however they do not take into account variabilities. [Dolan et al. 00] analyze quality attributes based on the SAAM method to enable comparison of competing PLA solutions. [Barbacci et al. 10] do not take into consideration variabilities to evaluate a PLA based on its quality attributes. SystEM-PLA does not use an ADL to correlate quality attributes and variabilities as there is no standardization among existing ADLs. SystEM-PLA aims to evaluate only one PLA solution at once by trading-off its quality attributes, in contrast to [Dolan et al. 00] approach. It uses variability-based scenarios to identify and analyze architectural problems, such as, variabilities in components that are not accurately identified and represented. Therefore, it takes advantage of such scenarios to support the definition of basic and quality attribute metrics to improve quantitative and qualitative analysis over the collected data. [Etxeberria and Sagardui 08] propose a feature model extension to evaluate the quality of a PLA based on its quality attributes. Such a work is limited to only qualitative analysis by not taking into consideration metrics to quantitatively measure PLA models and quality attributes.

PLA structural evaluation has been done by: [Hoek et al. 03], [Rahman 04], and [Kim et al. 08]. [Hoek et al. 03] present a set of metrics to evaluate a PLA with regard to its structural soundness quality attribute. [Rahman 04] proposes a set of metrics to evaluate the component structure of a PLA based on quality attributes. Both works propose metrics to support the evaluation of a PLA. However, they do not take into consideration PLA variabilities represented in PLA models to support quantitative analysis and improve qualitative analysis as it done in SystEM-PLA. This means that their work does not allow the analysis of PLA behavior based on variabilities which is interesting to analyze the PLA return on investment. [Kim et al. 08] present experiences with respect to PLA evaluation for the consumer electronics domain, based on the static implementation of product architectures and PLA variabilities, as well as the representation of quality attributes by using the Product Line Use Case (PLUC) tag.

Most of the work cited rely on the application or adoption of PLA evaluation techniques mainly focused on approaches that are able to draw qualitative analysis; an exception are the metric-related approaches. SystEM-PLA uses both qualitative and quantitative analysis.

Non-standard representation techniques [Kim et al. 08, Gannod and Lutz 00] are used in several approaches to state variabilities, scenarios and quality attributes; this makes it difficult to apply to PLs modeled in standard languages such as UML. SystEM-PLA instead is to be used with PLs modeled with UML.

3 The SystEM-PLA Method

SystEM-PLA aims to evaluate a PLA taking into account the variabilities represented in its models. It inherits most of its principles from ATAM, Holistic Method for Product Line Architecture Assessment (HoPLSAA), Extended ATAM (EATAM), and Goal-Question-Metric (GQM). Such techniques do not need any modification as they only serve as a basis to guide the SystEM-PLA planning phase proposition (Figure 1).

ATAM [Clements et al. 02] was chosen as it provides SystEM-PLA with guidelines to identify and define business drivers as well as performing tradeoff of quality attributes. The Software Architecture Analysis Method (SAAM) [Clements et al. 02] is not encompassed by SystEM-PLA as such a method is only focused on modifiability with extensions to testing and non-functional aspects. In addition, only ATAM can be exploited in order to incorporate variability analysis as the EATAM approach proposes. HoPLSAA [Olumofin 07] combined with EATAM [Kim et al. 08] provides directions on how to define quality attributes taking into consideration PL variabilities and how to analyze qualitatively variation points to perform PLA evaluations. EATAM provides

SystEM-PLA with variability scenarios for the support of trade-off analysis. GQM [Basili and Rombach 88] supports SystEM-PLA with questions, based on the PLA business drivers, for defining and collecting PLA quality attributes metrics. It also contributes to improve the PL metrics identification for an appropriated trade-off analysis. This approach provides a means to rationale the metrics defined for PL architecture evaluations based on the business goals of a PL, defined by the ATAM/EATAM approaches combination. Such metrics serve as input to data analysis. PLA quality attributes are analyzed to validate both the business drivers defined for a PLA and accuracy of modeled variabilities for a certain domain. In addition, SystEM-PLA can be used in a "what-if" basis to analyze design alternatives, by providing support to make decisions and analyze trade-offs that affect the PL products to be developed.

SystEM-PLA is considered as part of the PL development activities. It has three distinct phases as shown in Figure 1 - `Planning`, `Data Collection`, and `Data Analysis and Reporting` - which are supported by specific guidelines (Section 3.2). These phases are described as follows:

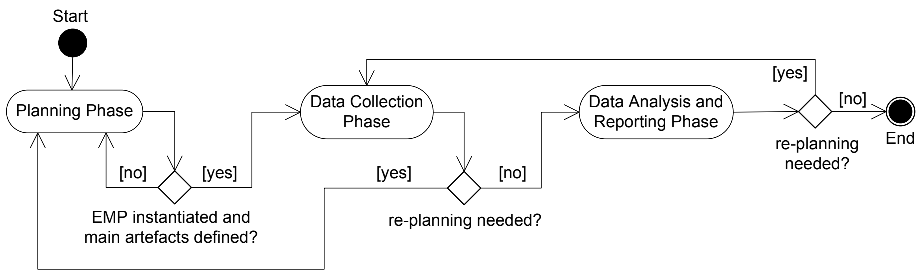


Figure 1: The SystEM-PLA Phases.

– The `Planning Phase` is concerned with instantiating the Evaluation Meta-Process (EMP) (Section 3.1). It has as inputs the feature model of the PL, as well as the PL models specified in UML including, at least, the class and component models of the PLA.

– The `Data Collection Phase` consists of conducting a PLA evaluation by generating PLA configurations. The basic metric suite (Section 3.3) and metrics defined for quality attributes are applied to these configurations to collect data. It has as input the defined EMP's artifacts (Section 3.1) for a PLA evaluation, and as outputs the generated PLA configurations and the data collected from metrics.

– The `Data Analysis and Reporting Phase` interprets in a qualitative and/ or quantitative basis in order to perform trade-off analysis and provide empirical evidences with regard to the PLA as well as prioritizing evaluated quality attributes.

3.1 The Evaluation Meta-Process (EMP)

EMP aims to define artifacts that will allow the execution of a PLA evaluation and the selection of the PLA quality attribute(s) to be evaluated; the definition of the managerial and technical questions to be answered with respect to the selected quality attributes; the definition of the quality attribute metrics to support the data collection and data analysis phases. The meta-process takes as input the logical view of a PLA, the PL feature model (FM) and its UML models. Figure 2 presents a UML activity diagram, which represents the EMP's activities (rounded rectangles) and their inputs and outputs (squared rectangles).

The following items present a brief description of each EMP's activity, which are performed based on the planning guidelines (Section 3.2.1):

Business Drivers Definition takes as input the `PL Models` and the PLA `Quality Attributes`, and defines the `Business Drivers` that a PLA should reach to develop its products. The defined business drivers support the definition of scenarios and managerial and technical questions. Although the business drivers definition activity is based on the ATAM method [Clements et al. 02], it is concerned with the PLA business drivers rather than the single product's architecture business drivers. It, therefore, requires using the modeled PL variabilities.

Scenarios Definition takes as input the `Business Drivers`, `Feature Model`, and PLA `Quality Attributes`. It generates the `Defined Scenarios` for each PLA quality attribute to support its selection activity.

Scenarios Ranking takes as input the `Defined Scenarios` in order to rank them based on PLA factors (Section 3.2.1). It generates as output `Ranked Scenarios`.

Scenario-based Quality Attributes Selection takes as input the `Ranked Scenarios` and selects which quality attributes will be evaluated for a certain PLA. Its output is a set of `Selected Quality Attributes`, which is a subset of the `Quality Attributes` set.

Managerial and Technical Questions Definition takes as input the `Business Drivers`, `Feature Model`, and `Selected Quality Attributes`. It defines the `Managerial and Technical Questions` that will be answered by defining metrics to support the data collection and analysis. Such questions are defined with regard to the PLA business drivers, and they take into account the roles involved in the PL process as in [Chastek and Ferguson 06].

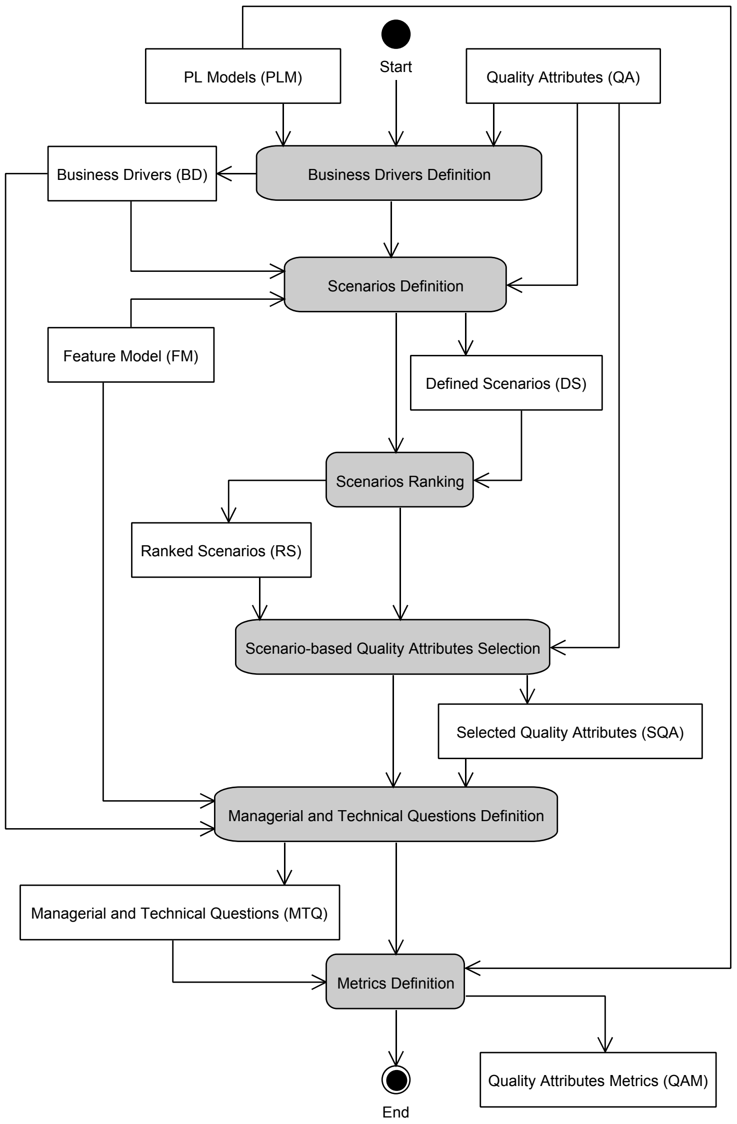


Figure 2: The Evaluation Meta-Process (EMP).

Metrics Definition: takes as input the `PL Models`, `Selected Quality Attributes` and the `Managerial and Technical Questions`. It defines `Quality Attributes Metrics` to answer such questions and to support data collection and quantitative analysis in PLA evaluations.

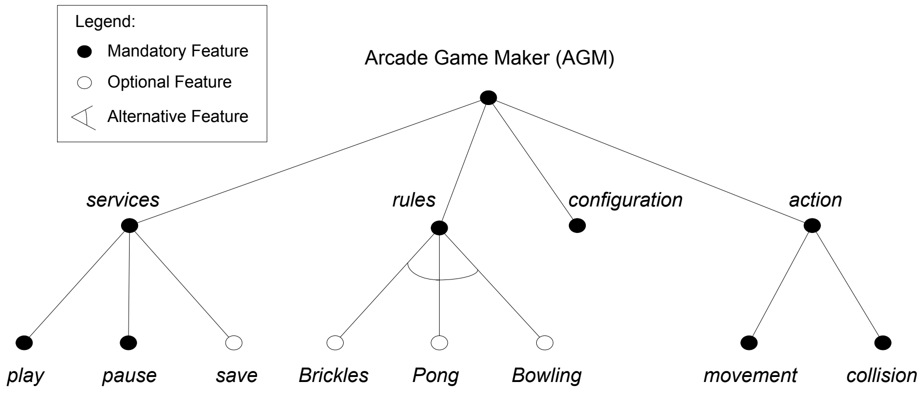


Figure 3: The Arcade Game Maker Feature Model.

3.2 SystEM-PLA Evaluation Guidelines

This section presents excerpts of the evaluation guidelines, which SystEM-PLA users must follow in order to properly perform PLA evaluations in a systematic way. Such guidelines are illustrated with the Arcade Game Maker (AGM) PL [SEI 10]. The AGM PL was created by the Software Engineering Institute (SEI) to support learning and experimenting based on PL concepts. It has a complete set of documents and UML models, as well as a set of tested classes and source code for three different games: Pong, Bowling, and Brickles. For the illustration purpose we only take into account complexity and extensibility quality attributes. However, the user can come up with additional quality attributes.

The essential AGM models are: the feature model, the use case model, the class model, and the component model. The **feature model**, presented in Figure 3, is concerned with four top-level features for AGM products, which are: `services`, `rules`, `configuration`, and `action`. The other models were adapted from the original SEI's models to represent variabilities by applying a set of stereotypes defined by the SMarty (**S**ystematic **M**anagement of V**ar**iabili**ty** in UML-based Software Product Lines) approach [Oliveira Junior et al. 10]. Basically, variation points are tagged with ≪`variationPoint`≫, and variants are tagged with one of the following: ≪`mandatory`≫, the variant appears in every PL product; ≪`alternative_OR`≫, one or more variants must be selected; ≪`alternative_XOR`≫, one variant must be selected; and ≪`optional`≫, a variant might be or not present in a PL product.

Use cases can be traced from the AGM features. The AGM **use case model** (Figure 4) has a mandatory variation point `Play Selected Game`. It has three inclusive variants, which are: `Play Brickles`, `Play Pong`, and `Play Bowling`. The AGM use case model also has two optional use cases `Check Previous Best`

Score and Save Score with respective variabilities. The selection of the former use case forces the selection of the latter due to the ≪requires≫ constraint. The other use cases and the actors are mandatory variants.

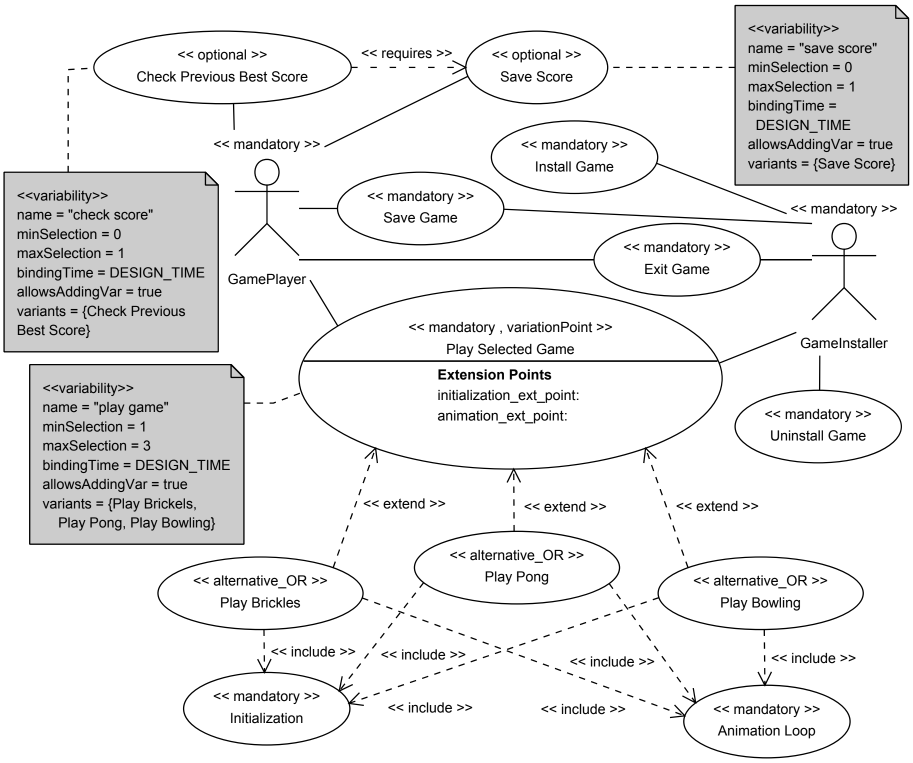
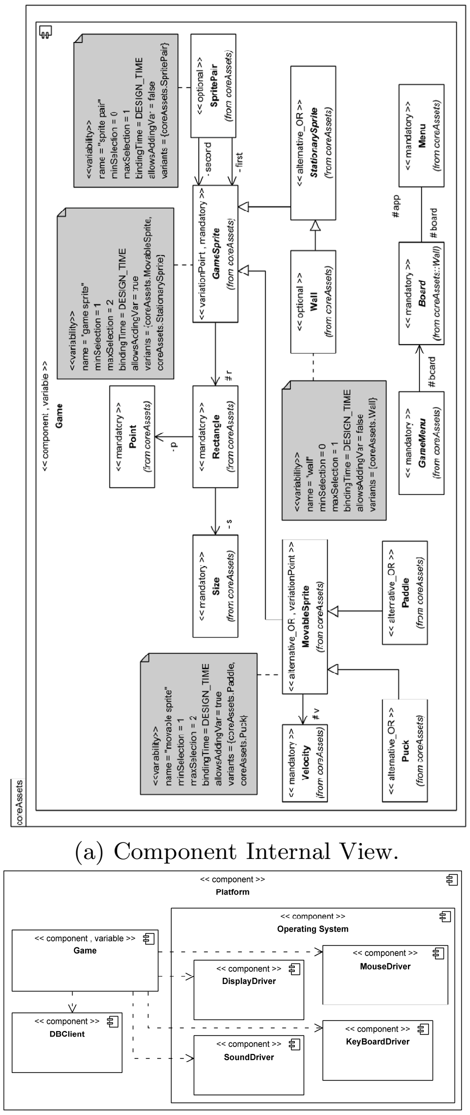


Figure 4: Arcade Game Maker Use Case Model and its Variabilities.

The AGM **class model** (Figure 5a) is traced from the realization of the AGM use cases. It has a mandatory variation point, the abstract class GameSprite, which has the alternative variants MovableSprite and StationarySprite. The former variant is also a variation point, which has the alternative variants Puck and Paddle. The AGM core asset class model also has two optional variants: Wall and SpritePair. The other classes are mandatory variants of the AGM core asset class model.

As the AGM core asset classes have explicit variabilities and they form the basis of the component Game in the AGM **component model** (Figure 5b), such a component is stereotyped as ≪variable≫. The other components of



(a) Component Internal View.

(b) Architecture Logical View.

Figure 5: Arcade Game Maker Structural Views.

the model are non-variable components, with no variability-concern stereotype indication.

3.2.1 Planning Guidelines

The following planning guidelines must be taken to perform each EMP activity (Section 3.1):

1. **Define business drivers (if it has not been done):** the user must define properly PLA business drivers by following his/her own strategy or based on the ATAM business driver definition as described by [Clements et al. 02] taking as inputs the PL models and the PL quality attributes (see Figure 2). For our AGM example, we defined the following business drivers based on [Olumofin 07] and [Kim et al. 08] experiences:
   - **BD.1 - keep game complexity degree lower than 0.7 (70%), compared to the overall PLA complexity, for at least 50% of produced products:** uphold low maintainability and low cost rates by focusing on complexity. Complexity degrees can provide an indicator of how difficult is to maintain the products derived from a PLA. Thus, the harder is to maintain a product, the higher is its cost.
   - **BD.2 - keep game extensibility degree higher than 0.75 (75%), compared to the overall PLA extensibility, for at least 50% of produced products:** maintain high reuse rate by focusing on extensibility. Extensibility factors can provide an indicator of how reusable is a product in terms of its components. The more extensible is a component, the higher is its reusability rate.
2. **Define scenarios for the quality attributes:** the user must define scenarios for each PLA quality attribute by following his/her own strategy or the ATAM scenarios definition activity as described by [Barbacci et al. 10] and [Clements et al. 02]. Such scenarios definition must take into account the feature model, defined business drivers, and PLA quality attributes. Each defined scenario must explicitly indicate the selected quality attribute it affects and the scenarios description. In addition, the scenarios must be represented in a utility tree to facilitate their prioritizing and presentation, respectively, the related top-level features, and sub-features. Features support the scenarios specification by linking a business driver to one or more scenarios. For instance, the AGM top-level features services, rules, and actions are related to the scenarios of the business driver BD.1. Table 1 and Table 2 present the utility trees for the complexity and extensibility defined scenarios.
3. **Rank scenarios:** the user must rank, as High (H), Medium (M), or Low (L), each defined scenario by taking into account the following concern attributes: its overall **importance** for the PLA and its business drivers; the **generality** of the scenario with respect to the PLA. It is ranked as mandatory (High),

Table 1: AGM Defined Scenarios for Complexity.

AGM - Quality Attribute Utility Tree		
Quality Attribute	*Complexity*	
Related Feature(s)	*services, rules, actions*	
Related Business Driver(s)	***BD.1:*** *keep game complexity degree lower than 0.7 (70%), compared to the overall PLA complexity, for at least 50% of produced products*	
Scenario(s)	Sc.1	Variation points and/or variants are added, modified, or removed maintaining the BD.1 true.
	Sc.2	50% of variabilities are removed maintaining the BD.1 true.
	Sc.3	One-game environments have complexity values at most 0.65 (65%) compared to the overall AGM PLA complexity.

Table 2: AGM Defined Scenarios for Extensibility.

AGM - Quality Attribute Utility Tree		
Quality Attribute	*Extensibility*	
Related Feature(s)	*services, rules, actions*	
Related Business Driver(s)	***BD.2:*** *keep game extensibility degree higher than 0.75 (75%), compared to the overall PLA extensibility, for at least 50% of produced products*	
Scenario(s)	Sc.4	Variation points and/or variants are added, modified, or removed maintaining the BD.2 true.
	Sc.5	50% of variabilities are removed maintaining the BD.2 true.
	Sc.6	Two-game environments have extensibility values at least 0.8 (80%) compared to the overall AGM PLA extensibility.

alternative (Medium), and optional (Low) as in [Olumofin 07]; its **cost/risk**, i.e., the effort involved in providing proper responses to the scenarios, as well as its perceived risk; and the **number of variability**, encompassed by a scenario.

Table 3 presents the complexity and extensibility quality attribute scenario ranking for our AGM illustration.

4. **Select quality attributes based on ranked scenarios:** as the scenarios are ranked, the user must select the quality attributes by defining a strategy. A widely known used strategy is the voting system as in the ATAM method [Barbacci et al. 10], [Clements et al. 02]. However, the user can define his/her own way to select the quality attributes to be evaluated, as long as the definition avoids quality attribute conflicts [Barbacci et al. 10] that might impair the evaluation results. For the AGM example, we selected the scenarios based on the following analysis:

   – scenarios Sc.1, Sc.4 and Sc.5 have high number of variability and overall

Table 3: AGM Complexity and Extensibility Scenarios Ranking.

Business Drivers			***BD.1***			***BD.2***		
Quality Attributes			***Complexity***			***Extensibility***		
Scenarios			***Sc.1***	***Sc.2***	***Sc.3***	***Sc.4***	***Sc.5***	***Sc.6***
Scenarios Ranking	**Overall Importance**	**H**	**X**		**X**	**X**	**X**	**X**
		M		**X**				
		L						
	Generality	**H**	**X**			**X**		
		M		**X**			**X**	
		L			**X**			**X**
	Cost/Risk	**H**		**X**			**X**	
		M	**X**			**X**		
		L			**X**			**X**
	Number of Variability	**H**	**X**	**X**		**X**	**X**	**X**
		M			**X**			
		L						

importance to the PLA. In addition, such scenarios are mandatory to the AGM PLA with medium cost/risk;

- scenario Sc.6 also has the same ranking for amount of variability and overall importance, as well as a low cost/risk and it is optional;
- scenario Sc.2 has a high amount of variability and cost/risk to the AGM PLA. In addition, it has a medium importance to the PLA and it is alternative; and
- scenario Sc.3 is alternative and it has high importance to the AGM PLA. It also has a low cost/risk and amount of variabilities.

Thus, scenarios Sc.1, Sc.4 and Sc.5 (Table 4) are the most important for the AGM PL. Sc.1 is related to the business driver BD.1, whereas Sc.4 and Sc.5 are related to the business driver BD.2. Therefore, complexity and extensibility quality attributes were selected. In a more complex example involving a larger set of quality attributes, the selected attributes are usually a subset of the initial set.

5. **Define managerial and technical questions:** the user must use the business drivers, feature model, and selected quality attributes to define the questions. Although one can use particular methods, we recommend the use of the GQM method, due to its maturity and consolidation. In SystEM-PLA, business drivers, quality attributes and features might represent the GQM goals and they are used to define the GQM questions. Such questions must

Table 4: AGM Quality Attributes Selection.

Business Driver	Quality Attribute	Scenario #	Selection Order
BD.1	Complexity	Sc.1	1st
BD.2	Extensibility	Sc.4	2nd
BD.2	Extensibility	Sc.5	3rd
BD.2	Extensibility	Sc.6	4th
BD.1	Complexity	Sc.2	5th
BD.1	Complexity	Sc.3	6th

indicate which business driver, quality attribute and/or feature they are related to. In addition, each question must have a unique identification, as well as a set of related GQM metrics. Thus, Table 5 presents the managerial and technical questions defined for the business drivers of the AGM PLA.

Table 5: AGM Managerial and Technical Questions for the PLA Business Drivers.

Business Driver / Feature / Selected QA (Goals)	Questions	
Low Complexity (business driver)	Q.01	What is the complexity of a class/interface in a class model?
	Q.02	What is the complexity of a variation point class/interface in a class model?
	Q.03	What is the complexity of a variability class/interface in a class model?
	Q.04	What is the complexity of a variable component in a model?
	Q.05	What is the complexity of a PLA based on its class model?
High Extensibility (business driver)	Q.06	What is the extensibility of a class/interface in a class model?
	Q.07	What is the extensibility of a variation point class/interface in a class model?
	Q.08	What is the extensibility of a variability class/interface in a class model?
	Q.09	What is the extensibility of a variable component in a component model?
	Q.10	What is the extensibility of a PLA based on its class model?

6. **Define the quality attribute metrics:** the user must take into consideration the selected quality attributes and the questions stated to define metrics for quality attributes. Such metrics answer the questions with regard to each selected PLA quality attribute. The basic metric suite (Section 3.3) might be used to compose the new metrics. Each metric must indicate which quality attribute it is related to, and the question it answers. For the AGM example, we defined twelve metrics, six to measure complexity and six to measure extensibility, showed in more details in Sections 3.4 and 3.5. Table 6 summarizes the defined metrics and their brief descriptions.

Complexity metrics from Table 6 were defined based on the *Weighted Methods per Class* (WMC) metric, which is the sum of the McCabe's *Cyclometic*

Table 6: AGM Metrics for Complexity and Extensibility Quality Attributes.

Quality Attribute	Question	Metric	
		Name	Description
Complexity	Q.01	CompInterface	Value of the Weighted Methods per Class (WMC) complexity metric for an interface. It is always 0.0 as interfaces do not have concrete methods.
		CompClass	Value of the Weighted Methods per Class (WMC) complexity metric for a class.
	Q.02	CompVarPointClass	Sum of the *CompClass* or *CompInterface* value of all its associated variants plus the *CompClass* or *CompInterface* value of the variation point.
	Q.03	CompVariabilityClass	Sum of *CompVarPointClass* associated with all variabilities in class models.
	Q.04	CompVarComponent	Sum of the *CompVariabilityClass* value for all the variabilities associated with classes that form a component.
	Q.05	CompPLA	Sum of the *CompVarComponent* value for all the components of a PLA.
Extensibility	Q.06	ExtensInterface	Number of abstract methods divided by the number of methods of an interface, i.e., it is always 1.0.
		ExtensClass	Number of abstract methods divided by the number of methods (concrete plus abstract) of a class.
	Q.07	ExtensVarPointClass	Value of the *ExtensClass* or *ExtensInterface* of an abstract class or interface multiplied by the number of its subclasses or implementation classes.
	Q.08	ExtensVariabilityClass	Sum of *ExtensVarPointClass* associated with all variabilities in class models.
	Q.09	ExtensVarComponent	Sum of the *ExtensVariabilityClass* value for all the variabilities associated with classes that form a component.
	Q.10	ExtensPLA	Sum of the *ExtensVarComponent* value for all the components in class models of a PLA.

Complexity (CC) (see Section 3.4).

To ensure the effectiveness of the defined metrics, the user must empirically validate them by means of an experimental study. Taking into account the complexity and extensibility metrics proposed in the AGM example, we validated them as an experiment [Oliveira Junior et al. 10] carried out including six subjects who had to generate configurations for the AGM PL to which the complexity and extensibility metrics were applied, collected, analyzed, and validated.

Figure 6 shows the GQM model for the AGM example resultant from the planning guidelines.

As we mentioned in Section 3.1, the post conditions of the planning phase are the instantiation of EMP and the definition of the main artifacts for PLA evaluations, including quality attribute metrics. Sections 3.4 and 3.5 present the formal definitions of the defined metrics for the AGM example. These metrics use some of SystEM-PLA's basic metrics in their definitions (Section 3.3).

3.2.2 Data Collection Guidelines

A PLA configuration, or product configuration, is an instance of the PLA, which represents a single-product architecture with most of the PLA variabilities resolved. Thus, the user can perform trade-off analyses of the PLA with respect to its products and its quality attributes.

The following items present the data collection guidelines:

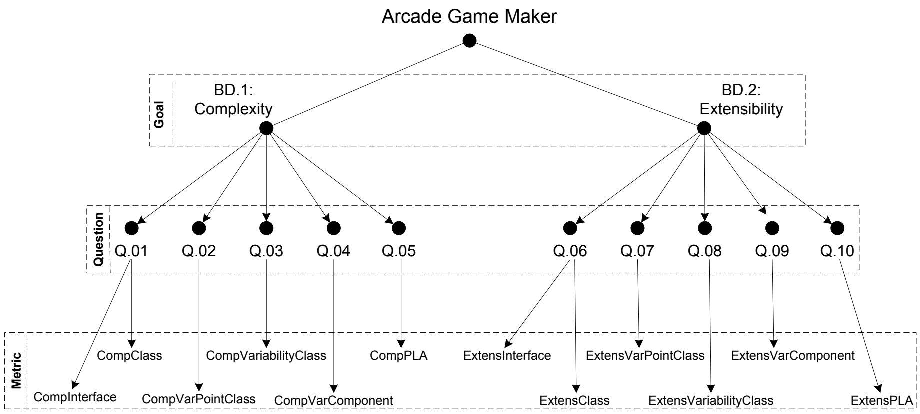


Figure 6: AGM Goal-Question-Metric Model.

1. **Create PLA configurations:** it can be done manually or automatically. The former is more complex and demands a set of people to carry it out as it requires much more attention on checking whether the created configurations are valid. The later is more reliable since one uses tools to generate valid configurations. We are currently generating our configurations manually as we are developing an experimental environment for PL evaluation in which such a generator is been built in order to make the generation step more reliable.

2. **Collect data by applying the defined quality attribute metrics:** it is done by calculating the quality attribute metrics applied to the created configurations of the PLA. This can be done manually or automated. As we are concerned about UML-based PLA, we recommend the use of an automated tool, such as [SDMetrics 10]. This kind of tool provides some features to define customized metrics and calculate them from UML modeling tools exported as XMI files. For the AGM example, we applied the metrics for complexity and extensibility to the AGM configurations created. Table 7 presents the observed values of the metrics *CompPLA* and *ExtensPLA* for each AGM configuration.

We carried out a study to empirically validate the metrics for complexity and extensibility quality attributes [Oliveira Junior et al. 10] presented in Sections 3.4 and 3.5. In this study, six participants manually created five valid and different AGM configurations by filling out a PLA template to resolve the AGM variabilities.

Table 7: Observed Values of Complexity and Extensibility Metrics for AGM Configurations.

Configuration #	CompPLA	ExtensPLA	Configuration #	CompPLA	ExtensPLA	Configuration #	CompPLA	ExtensPLA
1	0.51	0.61	11	0.53	0.61	21	0.49	0.61
2	0.56	0.61	12	0.97	1.00	22	1.00	1.00
3	0.51	0.81	13	0.48	0.61	23	0.52	0.61
4	0.83	0.80	14	0.69	0.61	24	0.42	0.61
5	0.91	1.00	15	0.74	0.80	25	0.62	0.80
6	0.50	0.61	16	0.98	1.00	26	0.47	0.61
7	0.47	0.61	17	0.77	0.80	27	0.53	0.61
8	0.53	0.61	18	0.82	0.80	28	0.70	0.80
9	0.67	0.80	19	0.52	0.61	29	0.40	0.61
10	0.90	1.00	20	0.82	0.80	30	0.78	0.80

3.2.3 Data Analysis and Reporting Guidelines

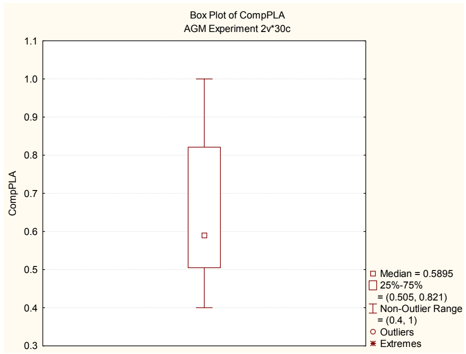
The data analysis is performed based on the artifacts produced by the previous SystEM-PLA phases.

Some of these artifacts lead to a **quantitative analysis**, such as: How many products are at most 15% less complex than the PLA itself? What is the impact, in terms of the extensibility degree, to the overall PLA by replacing some variation point abstract class with an interface? Some artifacts lead the user to a **qualitative analysis**, such as: Are the defined business drivers appropriated to the quality attributes of a PLA? If no, should we re-state some of these business drivers? Based on the observed values of the defined quality attribute metrics, can we say that the quality attribute “A” must be prioritized over the quality attribute “B” during the PL products development?

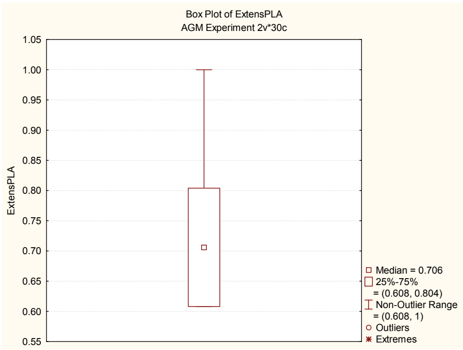
The following items present the data analysis and reporting guidelines:

1. **Plot the data in one or more graphical representations:** one can use different techniques to represent graphically the collected data from the previous phase. Such techniques might be: descriptive statistics and frequency distribution graph, which are important to statistical analysis; bar and pie charts as they provide information with regard to certain observed values over the whole measurement; and dispersion diagram, which is useful to compare the behavior of two different sets of measures plotted in the same graph. For the AGM example, the collected data was plotted in box plots, Figures 7a and 7b, and in a dispersion histogram, Figure 8.

2. **Analyze the descriptive statistics of the data:** it must be done based on some important statistical information with respect to the collected data, which are: the number of observed (measured) elements (N); the mean; standard deviation (StdDv), which shows how much variation there is from the mean; and median, which is the central numeric value separating the higher half of the observed set of values from the lower half. Thus, for the AGM example, we can observe that:



(a) Boxplot for CompPLA.



(b) Boxplot for ExtensPLA.

Figure 7: Collected Data Boxplots.

- **Analysis #1:** in Figure 7a for the CompPLA metric, the median value is 0.5895. This means that:
  - 15 (50%) configurations have CompPLA values less or equal to 0.5895;
  - 15 (50%) configurations have CompPLA value greater than 0.5895.
- **Analysis #2:** in Figure 7b for the ExtensPLA metric, the median value is 0.706. This means that:
  - 15 (50%) configurations have ExtensPLA value less or equal to 0.706;
  - 15 (50%) configurations have ExtensPLA value greater than 0.706.

3. **Identify how many scenarios satisfy the selected quality attributes:** based on the analysis performed over the descriptive statistics of the collected data, one can identify which scenarios previously stated satisfy the selected PLA quality attributes. This is essential to verify if either the scenarios are appropriated to the quality attributes or re-state them. We recommend that

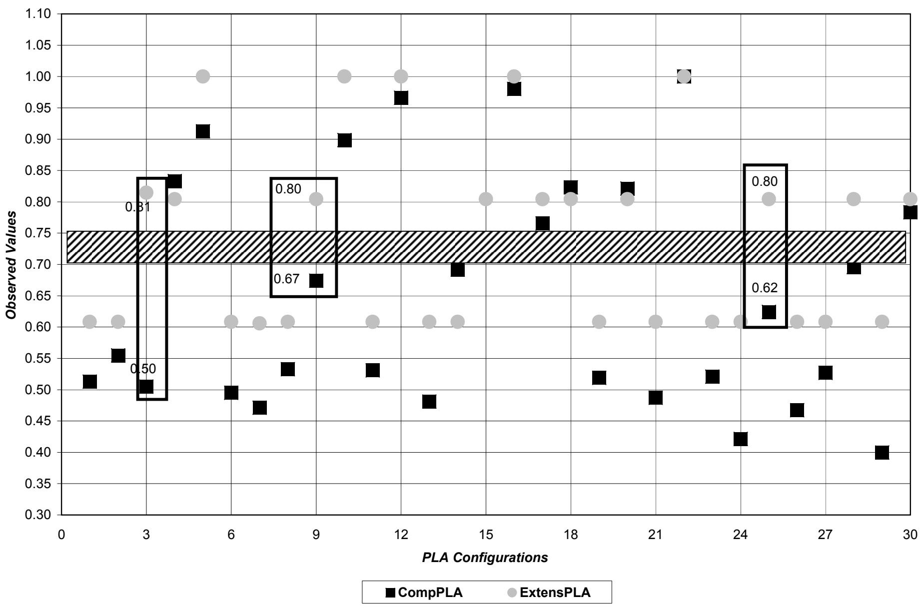


Figure 8: Dispersion Histogram of CompPLA and ExtensPLA Observed Values for the AGM Configurations.

at least 50% of the scenarios must satisfy the respective selected quality attributes. Otherwise, the scenarios do not provide a means to a reliable trade-off analysis. In the AGM example, the creation of the products only exercises scenarios Sc.1, for complexity, and Sc.4 and Sc.5, for extensibility. Thus, we can state that:

– **based on Analysis #1**, scenario Sc.1 (Table 1) is satisfied for the complexity quality attribute. During the creation of the AGM configurations, variation points and variants were modified or removed according to the sort of product created. Thus, scenario Sc.1 maintains the BD.1 true as 18 out of 30 configurations (60%) have CompPLA value less than 0.70 (see Table 7) as stated in such a scenario: "*keep game complexity degree lower than 0.7 (70%), compared to the overall PLA complexity, for at least 50% of produced products*";

– **based on Analysis #2**, scenarios Sc.4 and Sc.5 (Table 2) are satisfied for the extensibility quality attribute. Scenarios Sc.4 and Sc.5 maintain BD.2 true as 15 out of 30 configurations (50%) have ExtensPLA value greater than 0.75 (see Table 7) as stated in such scenarios: "*keep game extensibility degree higher than 0.75 (75%), compared to the overall PLA* *extensibility, for at least 50% of produced products*".

4. **Identify which selected quality attributes satisfy the PLA:** taking into account the percentage of scenarios that satisfy a quality attribute, we can determine which selected quality attributes satisfy the overall PLA. For the AGM example, both complexity and extensibility quality attributes satisfy the AGM PLA as 100% of their scenarios is satisfied.

5. **Perform a trade-off analysis:** it is done by taking into consideration the selected quality attributes that satisfy the overall PLA to decide which one(s) must be prioritized for the AGM products development and evolution. For the AGM example, this analysis was made by plotting the observed values of both CompPLA and ExtensPLA in a dispersion histogram (Figure 8). We can observe in this figure that the most interesting products are those which have values of CompPLA < 0.7 and ExtensPLA > 0.75. Thus, three main products become interesting for the AGM PL: the first one with CompPLA=0.50, and ExtensPLA=0.81; the second one with CompPLA=0.67, and ExtensPLA=0.80; and the third one with CompPLA=0.62, and ExtensPLA=0.80. Note that the ExtensPLA value for the three products is practically the same (0.80), which might be an indicator that for similar products we must prioritize complexity rather than extensibility. Another indicator might be the fact that 60% of the AGM products satisfy the complexity scenario Sc.1, whereas 50% of the AGM products satisfy the extensibility scenarios Sc.4 and Sc.5. The shaded region with values ranging from 0.7 to 0.75 contains unattractive products for the AGM PL as they do not conform to low complexity and high extensibility business drivers defined to the AGM PLA. We can conclude based on the two indicators that, for the AGM PLA, complexity must be more prioritized than extensibility. In case of none of the products become interesting for a PL, it must be analyzed which metrics values are out of the established range. Thus, it can be an indicator that the quality attributes related to such metrics should be prioritized or the business drivers re-stated.

6. **Write an evaluation final report:** to document all of the evaluation activities, as well as produced artifacts, strategies, collected data, and generated graphs and tables, we recommend the writing of an ATAM styled report [Clements et al. 02], as it is a well-known standard in the software architecture evaluation community. In addition, all the assets produced during the evaluation must be stored in order to allow the evaluation to be replicated in advance.

3.3 The Basic Metric Suite

SystEM-PLA provides a UML-based metric suite for measuring PL models [Oliveira Junior et al. 08]. The suite is defined based on the following PL models: actors, use cases, interfaces, classes, components, and diagrams.

The basic metric suite is composed of 98 metrics and can be used to compose new metrics for PLA quality attributes. An example of how to compose new metrics for the complexity quality attribute is presented in [Oliveira Junior et al. 08]. Although SystEM-PLA is focused on class and component models to evaluate a PLA, it also provides basic metrics for use cases and actors as we plan to expand our method to encompass high-level and behavioral models, as well as keep tracking of variabilities.

Each metric name is composed of five parts separated by an underline character ("_"). The **first part** is the acronym for the UML model that is being measured. The **second part** is the acronym of the UML model, used by the first part in its measurement. The **third part** is the acronym of the metric type. The **fourth part** is the acronym of the variability element. The **fifth part** is the acronym of the measure type. For instance, the metric `UCS_UCS_BAS_OPT_ISA` indicates whether a use case (UCS) is an (ISA) optional variant (OPT), and the metric `CLS_ITF_BAS_INC_NUM` measures the number (NUM) of alternative inclusive (INC) interfaces (ITF) associated with a class (CLS).

Tables 8 and 9 show the basic metrics for class and component models, diagrams, and overall PL. The basic metrics for actors and use cases are out of the scope of this paper.

3.4 Complexity Metrics

The analysis of PLA complexity, in this paper, has as a basis the [McCabe 76] Cyclomatic Complexity (CC) and Weighted Methods per Class (WMC) metrics. The CC is the measure of linearly independent paths of a source code, and the WMC is the sum of the CC of all methods for a class.

SystEM-PLA takes into account the variabilities modeled in class and component models, and the reasoning about main variabilities issues (Section 3.2) for PLA evaluation. Therefore, there were proposed five metrics for PLA complexity, which are: `CompClass`, `CompVarPointClass`, `CompVariabilityClass`, `CompVarComponent`, and `CompPLA`. The following items present their formal definitions.

***CompClass*:** is the WMC value for a given class. Interfaces always have zero (0) for the WMC value. It is defined as follows:

$$CompClass = \begin{cases} 0 & \text{for interfaces} \\ WMC & \text{for classes} \end{cases}$$

Table 8: SystEM-PLA Basic Metrics for Class Models.

Ord.	Metric Name	Metric Description
01	ITF_ITF_BAS_VPT_ISA	Interface is a variation point
02	ITF_ITF_BAS_INC_ISA	Interface is an alt. incl. variant
03	ITF_ITF_BAS_EXC_ISA	Interface is an alt. excl. variant
04	ITF_ITF_BAS_OPT_ISA	Interface is an optional variant
05	ITF_ITF_BAS_MND_ISA	Interface is a mandatory variant
06	ITF_ITF_BAS_INC_NUM	Qty. of alt. incl. variants of an interface
07	ITF_ITF_BAS_EXC_NUM	Qty. of alt. excl. variants of an interface
08	ITF_ITF_BAS_OPT_NUM	Qty. of optional variants of an interface
09	ITF_ITF_BAS_MND_NUM	Qty. of mandatory variants of an interface
10	ITF_ITF_BAS_VPT_NUM	Qty. of var. points of an interface
11	ITF_CLS_BAS_INC_NUM	Qty. of alt. incl. classes of an interface
12	ITF_CLS_BAS_EXC_NUM	Qty. of alt. excl. classes of an interface
13	ITF_CLS_BAS_OPT_NUM	Qty. of optional classes of an interface
14	ITF_CLS_BAS_MND_NUM	Qty. of mandatory classes of an interface
15	ITF_CLS_BAS_VPT_NUM	Qty. of var. point classes of an interface
16	CLS_CLS_BAS_VPT_ISA	Class is a variation point
17	CLS_CLS_BAS_INC_ISA	Class is an alt. incl. variant
18	CLS_CLS_BAS_EXC_ISA	Class is an alt. excl. variant
19	CLS_CLS_BAS_OPT_ISA	Class is an optional variant
20	CLS_CLS_BAS_MND_ISA	Class is a mandatory variant
21	CLS_CLS_BAS_INC_NUM	Qty. of alt. incl. variants of a class
22	CLS_CLS_BAS_EXC_NUM	Qty. of alt. excl. variants of a class
23	CLS_CLS_BAS_OPT_NUM	Qty. of optional variants of a class
24	CLS_CLS_BAS_MND_NUM	Qty. of mandatory variants of a class
25	CLS_CLS_BAS_VPT_NUM	Qty. of var. points of a class
26	CLS_ITF_BAS_INC_NUM	Qty. of alt. incl. interfaces of a class
27	CLS_ITF_BAS_EXC_NUM	Qty. of alt. excl. interfaces of a class
28	CLS_ITF_BAS_OPT_NUM	Qty. of optional interfaces of a class
29	CLS_ITF_BAS_MND_NUM	Qty. of mandatory interfaces of a class
30	CLS_ITF_BAS_VPT_NUM	Qty. of var. point interfaces of a class

***CompVarPointClass*:** is the *CompClass* value for a given class that is a variation point, plus the sum of the *CompClass* values of its variants. It is defined as follows:

$$CompVarPointClass = CompClass_i + \sum_{j=1}^{nVariants} CompClass_j \text{ , where:}$$

i is a variation point class

$nVariants = CLS\_CLS\_BAS\_INC\_NUM + CLS\_CLS\_BAS\_EXC\_NUM + CLS\_CLS\_BAS\_OPT\_NUM$

***CompVariabilityClass*:** is the sum of the *CompVarPointClass* values of all variation points with which a variability is associated. It is defined as follows:

Table 9: SystEM-PLA Basic Metrics for Component, Diagrams, and Overall PL.

Ord.	Metric Name	Metric Description
01	CPT_CPT_BAS_VTN_HAS	Component has variation
02	CPT_CLS_BAS_VBT_NUM	Qty. of variabilities in classes of a component
03	DGM_ACT_BAS_VPT_TOT	Total var. point actors in a use case diagram
04	DGM_ACT_BAS_INC_TOT	Total alt. incl. actors in a use case diagram
05	DGM_ACT_BAS_EXC_TOT	Total alt. excl. actors in a use case diagram
06	DGM_ACT_BAS_OPT_TOT	Total optional actors in a use case diagram
07	DGM_ACT_BAS_MND_TOT	Total mandatory actors in a use case diagram
08	DGM_ACT_BAS_VPT_TOT	Total var. point actors in a use case diagram
09	DGM_UCS_BAS_VPT_TOT	Total var. point use cases in a use case diagram
10	DGM_UCS_BAS_INC_TOT	Total alt. incl. use cases in a use case diagram
11	DGM_UCS_BAS_EXC_TOT	Total alt. excl. use cases in a use case diagram
12	DGM_UCS_BAS_OPT_TOT	Total optional use cases in a use case diagram
13	DGM_UCS_BAS_MND_TOT	Total mandatory use cases in a use case diagram
14	DGM_UCS_BAS_VPT_TOT	Total var. point use cases in a use case diagram
15	DGM_ITF_BAS_VPT_TOT	Total var. point interfaces in a class diagram
16	DGM_ITF_BAS_INC_TOT	Total alt. incl. interfaces in a class diagram
17	DGM_ITF_BAS_EXC_TOT	Total alt. excl. interfaces in a class diagram
18	DGM_ITF_BAS_OPT_TOT	Total optional interfaces in a class diagram
19	DGM_ITF_BAS_MND_TOT	Total mandatory interfaces in a class diagram
20	DGM_ITF_BAS_VPT_TOT	Total var. point interfaces in a class diagram
21	DGM_CLS_BAS_VPT_TOT	Total var. point classes in a class diagram
22	DGM_CLS_BAS_INC_TOT	Total alt. incl. classes in a class diagram
23	DGM_CLS_BAS_EXC_TOT	Total alt. excl. classes in a class diagram
24	DGM_CLS_BAS_OPT_TOT	Total optional classes in a class diagram
25	DGM_CLS_BAS_MND_TOT	Total mandatory classes in a class diagram
26	DGM_CLS_BAS_VPT_TOT	Total var. point classes in a class diagram
27	DGM_CPT_BAS_VBT_TOT	Total variabilities in a component diagram
28	MDL_ACT_BAS_VPT_TOT	Total var. point actors of a product line
29	MDL_UCS_BAS_VPT_TOT	Total var. point use cases of a product line
30	MDL_ITF_BAS_VPT_TOT	Total var. point interfaces of a product line
31	MDL_CLS_BAS_VPT_TOT	Total var. point classes of a product line
32	MDL_MDL_BAS_VPT_TOT	Total var. points of a product line
33	MDL_ACT_BAS_VBT_TOT	Total variabilities in actors of a product line
34	MDL_UCS_BAS_VBT_TOT	Total variabilities in use cases of a product line
35	MDL_ITF_BAS_VBT_TOT	Total variabilities in interfaces of a product line
36	MDL_CLS_BAS_VBT_TOT	Total variabilities in classes of a product line
37	MDL_MDL_BAS_VBT_TOT	Total variabilities of a product line
38	MDL_CPT_BAS_VTN_TOT	Total variable components of a product line

$$CompVariabilityClass = \sum_{i=1}^{nAssVP} CompVarPointClass_i \text{ , where:}$$

$$nAssVP = \text{DGM\_ITF\_BAS\_VPT\_TOT} + \text{DGM\_CLS\_BAS\_VPT\_TOT}$$

***CompVarComponent*:** is the sum of the *CompVariabilityClass* values of all variabilities in classes of a given component. It is defined as follows:

$$CompVarComponent = \sum_{i=1}^{CPT\_CLS\_BAS\_VBT\_NUM} CompVariabilityClass_i$$

***CompPLA*:** is the sum of the *CompVarComponent* values of all variable components of a PLA. It is defined as follows:

$$CompPLA = \sum_{i=1}^{MDL\_CPT\_BAS\_VTN\_TOT} CompVarComponent_i$$

3.5 Extensibility Metrics

Extensibility is a quality property in which simple changes to the design of a software artifact require a proportionally simple effort to modify its structure and source-code [Batory et al. 02]. Extensibility provides a means to add new functionalities to the designed software by exploiting its structure in terms of reuse. The PL approach makes extensibility possible by providing anticipated variability management.

One of the most important concepts with relation to object-oriented systems is **generalization**. It allows systems specialization in terms of their concrete classes implementation [Batory et al. 02, Nystrom et al. 04]. However, generalization affects the overall system structure. It usually requires the addition of more specialized classes to such a structure making systems domain-specific.

In order to avoid systems structure issues, with respect to generalization, we can exploit the **abstract classes** concept [Sane and Birchenough 99, Woolf 97]. An abstract class has a standard behavior, represented by a set of concrete methods. It can also be composed of abstract methods, which must be implemented by its first-level concrete subclasses. Abstract classes represent systems extension points by providing a means to extend the systems functionalities. Furthermore, they promote the **program to interface** concept [Nystrom et al. 04] which aims at developing abstract classes, and at making systems programming reliant on abstract rather than on concrete types. It also increases the number of extension points of a system and its extensibility, and it decreases the system structural impact. Such a concept is used as a basis to develop application frameworks [Sane and Birchenough 99].

Therefore, extensibility metrics take into account the following class relationships [OMG 10]: **generalization (inheritance)**, in which the general classifiers (superclasses) are the variation points and the specific classifiers (subclasses) are the variants; **interface realization**, in which the suppliers (specifications) are variation points and the implementations (clients) are the variants; **aggregation association**, in which the typed instances with hollow diamonds (shared association representation) are the variation points and the associated typed instances are the variants; and **composite aggregation**, in which the typed instances with filled in diamonds (composite association representation) are the variation points and the associated typed instances are the variants.

Based on such relationships, the following items present SystEM-PLA extensibility metrics:

***ExtensClassLevel*:** is the class extensibility level. It provides the percentage of abstract methods with relation to the total methods (abstract plus concrete) of a class. It is defined as follows:

$$ExtensClassLevel = \frac{CLS\_CLS\_EXT\_ABM\_NUM}{CLS\_CLS\_EXT\_MTD\_NUM + CLS\_CLS\_EXT\_ABM\_NUM}$$

***ExtensInterfaceLevel*:** is the interface extensibility level, which has always the value 1.0 as it is 100% composed of abstract methods. It is defined as follows:

$$ExtensInterfaceLevel = \frac{ITF\_ITF\_EXT\_MTD\_NUM}{ITF\_ITF\_EXT\_MTD\_NUM} = 1.0$$

***ExtensVarPointClassLevel*:** is the *ExtensClassLevel* value of the variation point class multiplied by the number of its variants. It is defined as follows:

$$ExtensVarPointClassLevel = \begin{cases} ExtensInterfaceLevel * nVarI \\ ExtensClassLevel * nVarC \end{cases}$$

where:

$nVarI = ITF\_ITF\_BAS\_INC\_NUM + ITF\_ITF\_BAS\_EXC\_NUM + ITF\_ITF\_BAS\_OPT\_NUM$
$nVarC = CLS\_CLS\_BAS\_INC\_NUM + CLS\_CLS\_BAS\_EXC\_NUM + CLS\_CLS\_BAS\_OPT\_NUM$

***ExtensVariabilityClassLevel*:** is the sum of the *ExtensVarPointClassLevel* values of all variation points with which a variability is associated. It is defined as follows:

$$ExtensVariabilityClassLevel = \sum_{i=1}^{nAssVP} ExtensVarPointClassLevel_i$$

where:

$nAssVP = DGM\_ITF\_BAS\_VPT\_TOT + DGM\_CLS\_BAS\_VPT\_TOT$

***ExtensVarComponentLevel*:** is the sum of the *ExtensVariabilityClassLevel* values of all variabilities in classes of a given component. It is defined as follows:

$$ExtensVarComponentLevel = \sum_{i=1}^{CPT\_CLS\_BAS\_VBT\_NUM} ExtensVariabilityClassLevel_i$$

***ExtensPLA*:** is the sum of the *ExtensVarComponentLevel* values of all variable components of a PLA. It is defined as follows:

$$ExtensPLA = \sum_{i=1}^{MDL\_CPT\_BAS\_VTN\_TOT} ExtensVarComponentLevel_i$$

4 Conclusion and Future Work

This paper presents SystEM-PLA, a systematic method to evaluate UML-based PLAs taking into account the PL variabilities, represented in UML models, which is domain-independent. SystEM-PLA encompasses a set of guidelines and basic UML metrics, which aim to provide directions to evaluators on how to plan, conduct, interpret results, and document PLA evaluations based on metrics for quality attributes. SystEM-PLA also provides a means to make PLA evaluations flexible by instantiating the evaluation meta-process. This allows the evaluator to define his/her own specific quality attributes, metrics, and techniques to define the EMP's essential artifacts for PLA evaluations. We used the SEI's Arcade Game Maker PL as proof of concept and illustrate how to follow SystEM-PLA guidelines to perform PLA evaluations.

While most of the current literature is mainly focused on specific PL and PLA evaluation approaches, our method is defined towards a flexible and general PLA evaluation approach allowing both quantitative and qualitative analysis of PLA quality attributes. Our method provides a means to perform trade-off analysis, as well as empirical analysis of collected data. In addition, SystEM-PLA can be used as a "what-if" way to analyze design alternatives by providing a means to make decisions and to analyze trade-offs that affect the products to be generated. SystEM-PLA also supports the planning, conducting, result analysis, and replication of experiments based on the collected data from PLA evaluations.

Current literature claims the need of PLA evaluation approaches; these would allow PL architects to analyze empirically the potential of a PLA, and would allow PL managers to analyze the aggregated managerial and economical values of a PL throughout its products. We showed that performing empirical feasibility analysis is essential to demonstrate the practical usefulness of a new method. SystEM-PLA's feasibility was empirically analyzed by having three professionals applying it in a large company. Although we have used a small PL to conduct our experiment, we had evidence that SystEM-PLA is feasible in industry and it can serve as a basis to analyze quantitatively and qualitatively a PLA based on its UML models and variabilities.

Based on current results, some directions for future work and contributions are suggested: (i) UML-based metrics for self-adaptive software towards supporting the definition of portability and/or scalability metrics; (ii) metrics for portability and scalability quality attributes in order to improve our AGM example as a proof of concept; (iii) the extension of stereotypes and tagged values

[Oliveira Junior et al. 10] to encompass variability in UML interaction diagrams; (iv) an automated tool to support the realization of SystEM-PLA activities, as well as plan and execute formal experiments. In addition, changes on various issues can be made to improve the experiments with SystEM-PLA, including: (i) increase the derived PLA configurations sample size, which is important to stay closer to real projects and to improve and generalize the results; (ii) conduct experiments in a more controlled environment; (iii) deal with real data from larger PLs; and (iv) recruit more industrial-environment well-qualified subjects from the Software Engineering and Information Systems areas.

Acknowledgments

The authors would like to thank CAPES-Brazil for funding Edson's visiting scholar term at the University of Waterloo, Ontario, Canada.

References

[Barbacci et al. 10] Barbacci, M. R., Clements, P. Lattanze, A. Northrop, L. Wood, W.: Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionic Systems: a Case Study. Technical Report CMU/SEI-2003-TN-012, Software Engineering Institute (SEI) (2010)

[Basili and Rombach 88] Basili, V. R., Rombach, H. D.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, pp. 758-773. (1988)

[Batory et al. 02] Batory, D., Johnson, C., MacDonald, B., Heeder, D.: Achieving Extensibility Through Product-Lines and Domain-Specific Languages: a Case Study, ACM Transactions on Software Engineering Methodologies, pp. 191-214. (2002)

[Böckle et al. 04] Böckle, G., Clements, P., McGregor, J. D., Muthig, D., Schmid, k.: Calculating ROI for Software Product Lines, IEEE Software, pp. 23–31. (2004)

[Chastek and Ferguson 06] Chastek, G., Ferguson, R.: Toward Measures for Software Architectures. Technical Note CMU/SEI-2006-TN-013, Software Engineering Institute (SEI) (2006)

[Clements et al. 02] Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)

[Dolan et al. 00] Dolan. T., Weterings. R., Wortmann, J. C.: Stakeholder-Centric Assessment of Product Family Architecture. In: Proceedings of the International Workshop on Software Architectures for Product Families, pp. 225-245. (2000)

[Etxeberria and Sagardui 08] Etxeberria, L., Sagardui, G.: Variability Driven Quality Evaluation in Software Product Lines. In: Proceedings of the Software Product Line Conference, pp. 243-252. (2008)

[Gannod and Lutz 00] Gannod, G. C., Lutz, R. R.: An Approach to Architectural Analysis of Product Lines. In: Proceedings of the International Conference on Software Engineering, pp. 548-557. (2000)

[Hoek et al. 03] Hoek, A., Dincel, E., Medvidovic, N.: Using Service Utilization Metrics to Assess the Structure of Product Line Architectures. In: Proceedings of the International Symposium on Software Metrics. IEEE Computer Society, pp. 298-308 (2003)

[Kim et al. 08] Kim, T., Ko, I. Y., Kang, S. W., Lee, D. H.: Extending ATAM to Assess Product Line Architecture. In: Proceedings of the IEEE International Conference on Computer and Information Technology, pp. 790-797. (2008)

[Linden et al. 07] Linden, F. J. van der, Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer-Verlag, New York (2007)

[McCabe 76] McCabe, T. J.: A Complexity Measure. IEEE Transactions on Software Engineering, pp. 308–320. (1976)

[Nystrom et al. 04] Nystrom, N., Chong, S., Myers, A. C.: Scalable Extensibility via Nested Inheritance. In: Proceedings of the Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 99-115. (2004)

[Oliveira Junior et al. 10] Oliveira Junior, E. A., Gimenes, I. M. S., Maldonado, J. C.: Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures. In: Proceedings of the Fourth Brazilian Symposium on Software Components, Architectures, and Reuse, pp. 31-40. (2010)

[Oliveira Junior et al. 10] Oliveira Junior, E. A., Gimenes, I. M. S., Maldonado, J. C.: Systematic Management of Variability in UML-based Software Product Lines, Journal of Universal Computer Science (J.UCS), pp. 2374-2393. (2010)

[Oliveira Junior et al. 08] Oliveira Junior, E. A., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In: Proceedings of the Conferencia Latinoamericana de Informática, pp. 489-498. (2008)

[Olumofin 07] Olumofin, F.: A Holistic Method for Assessing Software Product Line Architectures. VDM Verlag, Saarbrücken, Germany (2007)

[OMG 10] OMG - Unified Modeling Language (UML) v.2.2, `http://www.omg.org/spec/UML/2.2`

[Rahman 04] Rahman, A.: Metrics for the Structural Assessment of Product Line Architecture. Master Dissertation, School of Engineering - Blekinge Institute of Technology, Sweden (2004)

[Riva and Rosso 03] Riva, C., Rosso, C.: Experiences with Software Product Family Evolution. In: Proceedings of the International Workshop on Principles of Software Evolution, pp. 161-170. (2003)

[Pohl et al. 05] Pohl, K., Böckle, G., Linden, F. J. van der: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, New York (2005)

[Sane and Birchenough 99] Sane, A., Birchenough, A.: First Class Extensibility for UML - Packaging of Profiles, Stereotypes, Patterns. In: Proceedings of the International Conference on the Unified Modeling Language, pp. 265-277. (1999)

[SDMetrics 10] SDMetrics: The UML Design Quality Metrics Tool, `http://www.sdmetrics.com` (2010)

[SEI-HOF 10] SEI - Hall of Fame, `http://splc.net/fame.html`

[SEI 10] SEI - A Framework for Software Product Line Practice, `http://www.sei.cmu.edu/productlines/frame_report/index.html`

[SEI 10] SEI - Arcade Game Maker Pedagogical Product Line, `http://www.sei.cmu.edu/productlines/ppl`

[Taylor et al. 09] Taylor, R. N., Medvidovic, N., Dashofy, E. M.: Software Architecture: Foundations, Theory, and Practice. John Wiley & Sons, USA (2009)

[Woolf 97] Woolf, B.: The Abstract Class Pattern. In: Proceedings of the Pattern Languages of Programming Conference, pp. 1-8. (1997)