Abstract—A product-line architecture (PLA) entails a design reused by a family of products sharing several features. The long-term stability of a PLA largely depends on the modularization of each feature from the design outset. As many features of a product-line crosscut the PLA decomposition, their modularity and stability are often improved with aspect-orientation according to recent studies. However, the use of this technique in proactive design of PLAs is challenging and has rarely been investigated. The problem is that information about the crosscutting nature of features is limited in this case. This paper presents a first study aimed at analyzing the feasibility of deriving stable aspect-oriented PLAs through a proactive design method. Our study relies on a product line and presents an in-depth exploratory analyse. We compare the stability of aspect-oriented and alternative candidates for proactively-designed PLAs. The results indicate that it is possible to derive aspect-oriented PLAs with superior stability based on a proactive design method. Our study reveals that it is possible to anticipate pros and cons of PLAs with aspects when the latter are incorporated at an early design stage.

Keywords—Software Product Line Architecture; Aspect-Oriented Design; Software Architecture Measurement.

I. INTRODUCTION

It is often claimed that the use of aspects [7] can improve the stability [9], [10] of product-line architectures (PLAs) over time [6], [13]. However, the longevity of a PLA is largely dependent on ability of software architects to modularize crosscutting features [6], [13] from the design outset [6]. A feature is classified as crosscutting when it widely affects the modular decomposition of a PLA. Hence, a successful aspect-oriented (AO) design requires that architects come up with early decompositions that neatly modularize each product-line feature. This task is more challenging in the proactive design of AO PLAs, i.e. when it is only inferred from domain analysis artifacts, such as feature models. Thus, there is no previously-implmented PLA or product asset, from which developers can understand the crosscutting effect of features in a PLA decomposition [12].

Given the complexity of proactive designs of AO PLAs, software engineers need to be supported by architectural design methods (e.g. [4], [5], [13]). These methods are effective if benefits and drawbacks of derived architectures can be early revealed. Otherwise, software architects are often skeptical to adopt aspectual decompositions in proactive PLA engineering. This is particularly true if the impact of their AO architectural decisions cannot be contrasted with a conventional design alternative of the product line. Moreover, the inability to observe pros and cons of proactively-designed AO PLAs also hinders software architects in making more detailed design decisions [4].

Due to these difficulties, aspect-oriented PLA designs are not widely conceived in a proactive manner nowadays. On the contrary, aspects are often used in product line engineering in a very restrictive fashion: crosscutting features are observed in existing assets (e.g. object-oriented designs) and reactively extracted to aspects in product-line engineering. Moreover, all the studies assessing aspect-oriented product lines [6], [12], [13] focus on the analysis of decompositions that reactively departed from existing non-AO artefacts. In all these cases, detailed information about the crosscutting nature of features is available, thereby enabling designers to make more informed decisions on aspectual decompositions of product-line features. There is no work in the literature reporting to which extent proactive design methods can properly guide the derivation of aspect-oriented PLAs. As a consequence, little is known if the investment of applying aspectual decompositions upfront is worthwhile when compared with the reactive ‘aspectization’ of crosscutting features later.

This work presents an exploratory analysis of a PLA proactively designed with aspectual and alternative component-based decompositions. We compare the ability of these PLA decompositions remain stable and, therefore, become resilient to changes over time. UML Components [3] and extensions [4], [5], [11] comprised the proactive PLA...
design method employed in our study. UML Components is a consolidated and genuine component-based development method, which defines practical, simple and concise directives for proactive architectural design. Each of the target product lines, namely Arcade Game Marker [11] and Mobile Media [12], were developed following two candidate decompositions: an aspectual design and a pure component-based design (Section III). Our goal was to analyze if a comparison of such alternatives can enable architects to early identify benefits and drawbacks of AO PLAs (Section IV). Due to lack of space, information about the study related to Mobile Media was not included in this paper [1].

Our comparative analyses were driven by the application of architectural metrics that indicate several dimensions of component-level stability [9], [10] and feature-level stability [12] (Section II). We have learned the following lessons from the gathered results (Section IV):

1) The proactive AO design of AGM led to the derivation of architectures with superior feature stability when compared with their non-AO component-based counterparts. This finding was confirmed by an analysis of scattering [12], cohesion [12], and interaction [12] of features in both aspect-oriented and component-based alternatives. These results might indicate that it is possible to derive reasonably modular and stable AO PLA candidates in a proactive manner with a well-known component-based method;

2) The degree of feature interaction was lower in the proactively-designed AO PLA than in its pure component-based counterpart. This means that the AO architecture candidates avoid upfront undesirable side effects regarding the PLA stability, in particular when aspects are explicitly employed at early stages of product-line engineering.

II. PRODUCT-LINE ARCHITECTURES

The goal of PLA design is to maximize architecture reuse across several similar systems sharing common features [8]. A PLA defines the architectural elements required to realize feature commonality and feature variation across similar products. The architectural elements of PLAs studied in this paper follow component-based decompositions [2], that consist of components, interfaces, and their relationships.

The stability of a PLA can be predicted or assessed from different dimensions. First, properties of each architectural component and their dependencies need to be taken into account. These properties are extremely important to product-line engineering as they enable the design of stable architectures [9]. For instance, architectural components of a product line have to be minimally coupled to reduce the effort required for software maintenance [9]. Thus, changes in one component would have little or no impact on other components of the architecture. As a result, component-level metrics (Section II-A) can analyze the stability of any kind of software architecture, including PLAs.

However, these conventional component-level metrics may not be able to identify stability problems related to individual features or concerns in general [12]. As our focus on this paper is on feature-driven analysis, henceforward, we use the terms features and concerns interchangeably. Sant’Anna [12] proposed feature-driven architectural metrics to support the detection or prediction of architecture stability problems. Then, such feature metrics [12] (Section II-B) were used in our study to indicate the stability of each feature present in the PLA models. The feature-driven measurements allow the comparison of architecture design alternatives in terms of how well architecturally-relevant features are modularized and likely to be stable over time.

A. Component Stability Metrics

Component stability is a useful attribute to analyze the difficulty to change a software component. It is a characteristic that enables to indicate how much the component-based decomposition of an architectural design is rigid [9] to accommodate new changes. Conventional stability metrics proposed by Martin [9] are used in this work to measure different dimensions of component-level stability. Highly interdependent components lead to architectures that are rigid, non-reusable and hard to change. We performed a conceptual adaptation in the metrics’ definition in order to apply the stability metrics in the component-and-connector architecture. Instead of considering packages of classes (as prescribed in original metrics definition), here the conventional stability metrics are defined in terms of components.

The Martin’s metrics Afferent Coupling (Ca), Efferent Coupling (Ce) and Instability (I), defined in Table I are used to analyze the stability of components based on its dependencies. An architecture might have stable and unstable components. Stable components suffer few changes whereas unstable ones contain classes that are more likely to change. Abstractions may be used to extend functionalities without changing existing implementations. Thus, stable components have high abstraction whereas unstable components are highly concrete [9]. In this sense, Martin [9] defined the metric Abstraction (A) (Table I).

The metric Distance from the Main Sequence (D) [9] establishes a relation between metrics A and I (Table I). It measures how much a component is far from the main sequence, which represents the ideal balance between stability and abstraction. D values near to zero represent components that have a good balance between stability and abstraction in the component [9]. So, it is possible to design components more reusable and less sensitive to changes. Finally, the cohesion of architectural components is measured by the relational cohesion metric (H) [10]. Components are considered to have low cohesion when they exhibit few relationships between its internal elements.


Table I: Architectural Metrics Suites.

<table>
<thead>
<tr>
<th>Metrics Suite</th>
<th>Attribute</th>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>Component Dependency</td>
<td>Efferent Coupling (EC)</td>
<td>The number of classes inside a component that depend upon classes outside this component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Binary Coupling (BC)</td>
<td>The number of classes inside a component that depend upon classes outside this component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Independent Coupling (IC)</td>
<td>The number of classes inside a component that depend upon classes outside this component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abstraction (A)</td>
<td>It is calculated as the ratio of the number of abstract classes or interfaces in the component to the total number of classes and interfaces in the component. This metric ranges from 0 to 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distance of Main Suggestion (DMS)</td>
<td>It can be represented as the average number of external relationships per class in a component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relational Cohesion (H)</td>
<td>It can be represented as the average number of internal relationships per class in a component.</td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Feature Scattering</td>
<td>It counts the number of architectural components which contribute to the realization of a certain feature.</td>
</tr>
<tr>
<td></td>
<td>Stability</td>
<td>Feature Diffusion over Architectural Components (CDAC)</td>
<td>It counts the number of architectural components which contribute to the realization of a certain feature.</td>
</tr>
<tr>
<td></td>
<td>Metrics</td>
<td>Feature Diffusion over Architectural Interfaces (CDAI)</td>
<td>It counts the number of interfaces in the system architecture which contribute to the realization of a certain feature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature Diffusion over Architectural Operations (CDAO)</td>
<td>It counts the number of operations in the system architecture which contribute to the realization of a certain feature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature Interaction</td>
<td>It counts the number of other concerns with which the assessed feature s share at least an interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Component-Level Interlacing Between Features (CIBC)</td>
<td>It counts the number of other concerns with which the assessed feature s share at least a component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interfaced Interlacing Between Features (ICIB)</td>
<td>It counts the number of other concerns with which the assessed feature s share at least an interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operations-Level Interlacing Between Features (OIBC)</td>
<td>It counts the number of other concerns with which the assessed feature s share at least an operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature-Based Cohesion</td>
<td>It counts the number of features addressed by the assessed component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lack of Feature-Based Cohesion (LAC)</td>
<td>It counts the number of features addressed by the assessed component.</td>
</tr>
</tbody>
</table>

Table II: Exploratory Study Steps.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Step</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Development of the component-based architecture</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>Assignment of features to architectural elements from the architecture developed in step 1-a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>Review of the AO architecture by an expert</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Collecting feature-driven and conventional stability metrics from the AO architecture</td>
</tr>
</tbody>
</table>

B. Feature Stability Metrics

In an architectural design, features can be realized by different architectural elements, such as components, interfaces and operations. Moreover, each architectural element can realize several features. Therefore, features can interact, for instance, when there is more than a feature realized by the same architectural element [12]. The suite of feature metrics [12] employed in our study quantify three characteristics: feature scattering, feature interaction, and feature-based cohesion. These metrics were chosen in our study as they were found to work as good predictors of feature-level stability in previous studies (e.g. [6], [12], [13]).

The feature scattering metrics quantify the number of architectural elements affected by the realization of each feature. This group of metrics assumes that a feature scattered in a large number of elements is harmful to modularity and stability. It includes the metrics: CDAC, CDAI and CDAO, briefly defined in Table I. The metrics for quantifying feature interaction aims at identifying dependencies caused by features that are not well modularized, and, as a consequence, do not have well defined boundaries [12]. It is composed by three metrics, CIBC, OOBC and IIBC, defined in Table I.

LCC assesses feature-based cohesion. A component that encompasses a large number of features is likely to be unstable and it may suffer from effects coming from changes related to any of the features realized by it [12].

III. STUDY SETTINGS

Our exploratory study involved the proactive conception and analysis of architectural designs for the product line AGM (Arcade Game Maker) [11]. Our study was performed in three main stages as shown in Table II; for each product line, all of the steps from Table II were performed.

Proactive Design of PLAs. As initial steps, the alternative PLAs of AGM were proactively designed using: (i) a well-known method for component-based architectures, named UML Components [3] (step 1-a), and (ii) an extension of UML Components to derive the AO architectural designs, named DSBC/A [5] (step 2-a), so that we compare it with its non-AO counterpart produced in step 1-a. From now on, the pure component-based version of the PLA is also called non-AO architecture. UML Components was the chosen method for proactive PLA design because it is a well-known method in the community and it has been applied in a wide range of application domains. It also provides a well-documented process for eliciting components and interfaces for each product-line feature.

More importantly, an aspect-oriented extension of UML Components (i.e. DSBC/A [5]) was available to support the conception of aspect-oriented component-based architectures. The availability of this extension was essential to support the comparison of AO and non-AO architectures proactively designed with similar guidelines and rationale. DSBC/A extends UML Components with activities specifically tailored to guide the modularization of crosscutting features. It also proposes the elaboration of complementary models to explicitly support the identification of potential aspectual elements in the architectural design.

In addition to these methods, we needed mechanisms to explicitly identify and represent variability in the PLAs being designed. In order to support these tasks, we relied on an existing approach, named SMarty [11], in the steps 1-a and 2-a. This approach was used because it was also built to be used in conjunction of UML Components and its extensions. It also provides a UML profile that can be easily adapted to any conventional model to represent and manage variabilities, as well as a process that guides the identification, delimitation and representation of PLA variabilities [4] following a component-based decomposition.
Feature Modularization and Analysis. In steps 1-b and 2-b, features are assigned to architectural elements of PLAs identified in steps 1-a and 2-a, respectively. Then, domain experts review the PLA with the assigned features in step c for both cases. Finally, the application of the conventional and feature-driven metrics on the component-based and AO PLAs was carried out in step d. The measurement process is automatically supported by the SDMetrics tool.

A. Arcade Game Maker (AGM) Architectures

AGM [11] is a product line created by the Software Engineering Institute (SEI) to be used for benchmarking and experimentation. It encompasses three arcade games: Bricks, Bowling, and Pong. Its main variations are [11]: (i) specific rules of those games; (ii) kind, number and behavior of elements; and (iii) physical environment where the games take place. There are some common rules, such as: (i) each game has a set of Sprites; (ii) each game has a set of rules; and (iii) every game involves movement of elements.

Variability Identification. Based on the variabilities of use cases and business type models for AGM, a component-based architecture was designed (Figure 1). The component-based PLA was generated based on the application of UML Components and business layers [3]. All variabilities of AGM were identified with the support of SMarty [11]; their representation rely, for instance, on the use of the stereotype "variable" in the architecture models (Figures 1 and 2). For the sake of illustration, those models in the figures represent partial, high-level views of the produced PLAs. The annotation of a component with this stereotype indicates that it is formed by a set of classes with variabilities. In addition, features are explicitly assigned to architectural elements using stereotypes. The explicit feature assignment was performed to facilitate their location and the automatization of the quantitative analysis process. Examples of feature assignment stereotypes are: "Business", "Persistence", and "Exception Handling".

The components with suffix Ctrl indicate the controller components that access the services available in the business layer. These components provide operations to the GUI and dialog layers. The business components have the suffix Mgr. Figure 1 presents the component-based architecture for AGM. It has two system components: GameCtrl and GameBoardCtrl. They use the provided interfaces for business components that run the AGM games. The major distinguishing characteristic of AO architecture models is the inclusion of aspectual components represented by the stereotype "aspect comp spec".

The Aspect-Oriented PLA. Figure 2 presents the AO architecture for AGM. The aspectual components encapsulate operations that crosscut several components in order to modularize crosscutting features. Each crosscutting relationship is represented by the stereotype "intercepts". The aspectual component PersistDataMgr in the AGM product-line encapsulates the persistence crosscutting operations, which are required in several parts of the architecture. The aspectual component ExceptionHandlingMgr is responsible for operations related to exception handling. The components PersistDataMgr and ExceptionHandlingMgr were revealed in the identification of transversal use cases present in AGM, as suggested by the application of the DSBC/A method.

An additional issue to observe in this architecture is the separation of operations into more than one interface of business components. The separation allows the isolation of operations to be crosscut by independent aspects. An example is the interface IInstallationMgt (component GameMgr) that has operations crosscut by the component ExceptionHandlingMgr.

IV. QUANTITATIVE ANALYSIS

This section presents the measurement results obtained by the application of the architectural metrics to the proactively-designed non-AO and AO architectures of AGM (Section III). Hereafter, we discuss how the proactive design decisions contributed to derivation of good-enough aspect-oriented PLAs when compared to the non-aspect-oriented PLA counterparts.

Tables IIIa and IIIb present the results obtained by applying the component stability metrics (Table I) to the AO and non-AO architectures of the AGM product line. In order to facilitate data analysis, the most interesting results are highlighted in bold in all tables presented in this section.

Superior Stability of the Aspect-Oriented PLA. Comparing the values obtained by the metric of Instability (I) for the AO and non-AO architectures (Tables IIIa and IIIb), we conclude that the components GameBoardMgr and GameMgr became more stable in the AO architecture. This conclusion is supported by the measures I that tended to zero as the Ca (afferent coupling) metric increased. These results mean that more classes external to the components relied on classes that were internal to other components. As a result, the number of architecturally-relevant dependencies between PLA components was higher in the non-AO PLA than in its AO counterpart. The component GameCtrl also became more unstable as the value of metric Ce (efferent coupling)
increased, indicating that more classes internal to GameCtrl began to depend on classes external to it. The variations in Ca, Ce and I are thanks to the improved modularity leveraged by the new interfaces created to subsume crosscutting operations. The components ExceptionHandlingMgr and PersistDataMgr exist only in the AO architecture since they are related to crosscutting features.

### Stability vs. Abstraction Tradeoffs

The metric Abstraction (A) for the components GameBoardMgr and GameMgr is higher for the AO architecture in relation to the similar components of the non-AO architecture. This occurred because the crosscut operations have been logically separated and grouped into specific interfaces to improve the architecture modularity. As a result, the number of abstract classes increased, making the metric values to increase as well.

The values obtained for the metric Distance of the Main Sequence (D) show that the components GameBoardMgr, GameMgr, and GameCtrl of the AO architecture distanced themselves more from the ideal value (zero) (Tables IIIa and IIIb). This discrepancy occurred because, although the measures I has been generally better for the AO architectures, the measures A has increased more than the measures I for these components. So, this phenomenon directly interfered in the values of the metric D.

### Component vs. Feature Cohesion

Both non-AO and AO versions of the AGM architecture exhibited cohesive components according to the metric H. Nevertheless, the component stability metrics do not identify dependencies between features in architectural decompositions [12]; hence, the results of metric H does not reveal whether the AO approach is more cohesive than the non-AO approach from the feature modularity point of view. On the other hand, as far as the feature stability metrics (Table I) are concerned, the results show significant differences in favor to the AO versions with respect to the features identified in the AGM. For instance, Table IVb presents the results for feature-based cohesion captured by the LCC metric.

The analysis of the LCC measures reveal that components of the non-AO architecture exhibit lack of modularity as each of them embody various (non-crosscutting and crosscutting) features. The feature tangling in those components makes them more prone to undergo unwanted side effects due to upcoming changes in features realized by them, thereby increasing the likelihood of architectural instability. The measures show that components in the AO architecture are associated only with each non-crosscutting and crosscutting features originally assigned to them; therefore, there are likely to be more resilient to changes and suffer less unwanted effects due to feature diffusion on component implementation. It is noteworthy that the components ExceptionHandlingMgr and PersistDataMgr have the value 0 for the non-AO architecture because these components do not exist in this architecture. It is noticeable that LCC decreased for all other components.

### Lower Feature Scattering in AO PLA

Table IVa presents the results for the metrics computing feature scattering and feature interaction. The analysis of feature scattering measures (CDAO, CDAI, and CDAC) indicates that the features related to Business, Exception Handling and Persistence are scattered throughout several elements of the...
non-AO architecture. These observations are supported by the results in Table IVa. The inferior modularity of such features implies that changes in one of them could affect several elements of the architecture and, therefore, severely impair the long-term PLA modularity and stability.

The feature scattering measures obtained for the Business feature show that values have not suffered relevant improvements in the AO architecture. This was expected as this feature was not treated as an aspect. As the two architectures have been designed considering two main layers (Business and System), the Business feature is also present in almost all components of the AO architecture as well. On the other hand, for the features Exception Handling and Persistence, the values obtained showed that they are not spread to other elements of the AO architecture, being restricted to their respective aspectual components.

**Stronger Feature Couplings in the non-AO PLA.** In the analysis of the feature interaction metrics (CIBC, IIBC, OOBC), the values obtained for the non-AO architecture show that all features identified in the system interact with at least other feature, as shown in Table IVa. Stronger feature interaction is often the cause of additional component dependencies that can lead to unwanted side effects, hindering the architecture modularity and stability. The AO decomposition of the PLA is not exposed to this problem as the features Exception Handling and Persistence have been properly modularized. The consequence is that there are no dependencies between them at all as well as no additional dependencies with the Business feature are present (when compared with the non-AO counterpart).

**V. CONCLUDING REMARKS**

This work presented an analysis of alternative architectures proactively conceived for the product line AGM. The findings of our comparative analyses provide interesting indicators that an existing architectural design method can be used with acceptable confidence to derive good-enough AO decompositions of a product line. These architectural designs can be used by architects to reveal potential benefits and drawbacks early on the design process. Then, they can also be used to enable designers make more detailed design decisions, while having some confidence that the key decomposition is modular and resilient to changes. We also observed that the benefits of aspect-orientation are likely to be increased when applied from the PLA design outset. As a future work, besides aspect-orientation other alternative techniques can be used to evolve PLAs, such as dependency injection and refactoring.

**ACKNOWLEDGMENTS**

This work was supported by Fundação Araucária.

**REFERENCES**