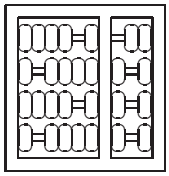

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Components meet Aspects: Assessing Design Stability of a Software Product Line

Leonardo P. Tizzei *Marcelo Dias*
Cecília M.F. Rubira *Alessandro Garcia*
Jaejoon Lee

Technical Report - IC-09-25 - Relatório Técnico

July - 2009 - Julho

Components meet Aspects: Assessing Design Stability of a Software Product Line

Leonardo P. Tizzei* Marcelo Dias† Cecília M.F. Rubira‡
Alessandro Garcia § Jaejoon Lee ¶

Abstract

A Product Line Architecture (PLA) should remain stable accommodating evolutionary changes of stakeholder's requirements. Otherwise, architectural modifications may have to be propagated to products of a product line, thereby increasing maintenance costs. Hence, it is important to understand which techniques better cope with PLA stability through evolution. This paper presents a comparative study to evaluate the positive and negative change impact on PLA designs based on components and aspects. The objective of the evaluation is to assess when aspects and components promote PLA stability in the presence of various types of change. To support a broader analysis, we compare the stability of the joint application of components and aspects to a PLA design against the isolated use of aspect-oriented, object-oriented, and component-based design techniques. The results show that the combination of aspects and components tends to promote superior PLA resilience than the other PLAs in most of the circumstances.

1 Introduction

Software product line (SPL) engineering aims to improve development efficiency for families of software systems in a given domain. This aim at facilitating large-scale reuse through a Product Line Architecture (PLA) that is common to a variety of similar products in terms of their architectural elements [5][7].

The combination of SPL and Component-based Development (CBD) is a well-known technique to rapidly and efficiently derive products from a SPL [4]. In the CBD, software systems are developed composing interoperable and reusable blocks called *software components* [23]. Moreover, software components explicitly expose their services as provided interfaces and their dependencies as required interfaces. A component-based PLA fosters explicit representation of component specification and contributes to reduce coupling and increase cohesion, thereby improving SPL modularity and evolvability.

*Institute of Computing, University of Campinas, Brazil
†Institute of Computing, University of Campinas, Brazil
‡Institute of Computing, University of Campinas, Brazil
§Pontifical Catholic University of Rio de Janeiro, Brazil
¶Computing Department, Lancaster University, United Kingdom

It is known that software systems evolve, otherwise their use become progressively less satisfactory [17]. In particular, in the context of PLA, there are several different types of evolutionary changes, such as: introduction and removal of features, and changing a mandatory feature optional and vice-versa. They can potentially end up with architecturally-significant changes and thus causing high maintenance costs for the products. As such, it is important for organisations to achieve a controlled evolution and a stable PLA lies at the heart of it

The reuse of a PLA, among other factors, depends on the ability of its design to remain stable [19]. A *stable PLA* means that it can endure evolutionary changes by sustaining its modularity properties [9]. One of the promising approaches to support enhanced modularity is Aspect-Oriented (AO) programming [15] and some works advocate that AO programming improves PLA stability [3][9][21]. On the other hand, there is also initial evidence that the isolate use of aspect-oriented decompositions can lead to modularity instabilities in various circumstances [9][20]. Yet, there is a lack of studies assessing whether an AO approach combined with a component-based PLA can reduce these modularity instabilities.

This paper presents a comparative study to evaluate the positive and negative impact of software evolution on PLAs. The objective of its evaluation is to quantitatively and qualitatively assess to what extent the synergistic use of component and aspect promotes architecture design stability in the presence of various types of change. We evaluate the PLA stability of a hybrid approach (i.e. combined application of aspects and components) against the isolated use of component-based, OO, and AO approaches.

Our investigation focuses on eight releases of a SPL called MobileMedia, which were implemented using Java and AspectJ. Four different versions of MobileMedia product line were involved in our empirical study: (i) an OO version, (ii) an AO version, (iii) a component-based version, and (iv) a hybrid version where both component and aspects are employed. We used conventional metrics suites based on change impact [22] and modularity [24] for the architecture stability evaluation of the four versions. We observed that in the hybrid approach provided a more stable PLA in most of the eight releases. In this paper, we have adopted a component implementation model called COSMOS* [13], which can explicit map the architectural components and connectors of a PLA to source code and also contributes to achieve a modular design by implementing explicit provided and required component interfaces.

This paper is organized as follows: Section 2 presents some concepts about COSMOS* implementation model and aspects and PLA. Section 3 describes the empirical study, which provides data for the change impact analysis on Section 4 and for the modularity analysis on Section 5. Section 6 presents some works related to this one and in Section 7 we draw the conclusions and plan the future work.

2 Background

2.1 COSMOS* Component Implementation Model

The main advantages of COSMOS*, when compared with other component models such as Corba Component Model (CCM)[2] and Enterprise Java Beans, is twofold. First, COSMOS* explicitly represents architectural units, such as components, connectors and configuration,

thus providing traceability between the software architecture and the respective source code. Second, COSMOS* is considered a platform-independent model, since it is based on a set of design patterns. Based on these advantages, we have chosen the COSMOS* implementation model to realize the component-based PLAs involved in our empirical study. However, our proposal could be applied to other component models, such as CCM[2] and Enterprise Java Beans.

COSMOS* defines five sub-models, which address different aspects of component-based systems: (i) the specification model specifies the components using UML; (ii) the implementation model explicitly separates the definition of the provided and required interfaces of the components from the implementation of its provided services; (iii) the connector model specifies the link between components using connectors, thus enabling two or more components to be connected in a configuration; (iv) composite components model specifies high-granularity components, which are composed by other COSMOS* components; and (v) system model defines a software component which can be executed straightforward, thus encapsulating the necessary dependencies.

2.2 Aspects and Product Line Architectures

Aspect-Oriented Software Development (AOSD) [11] is an approach that aims to modularize the crosscutting concerns of both standalone applications and software product lines. These concerns are widely-scoped properties and usually crosscut several modules in the software system. Aspects are the abstractions used to encapsulate otherwise crosscutting concerns. As advocated by some authors [3][9][21] AO design decompositions is a promising approach to support modularity and stability of crosscutting features in SPLs. The use of aspects rely on three majors mechanisms to modularize and vary crosscutting concerns (in this paper, concerns are considered equivalent to features): (i) joinpoints are the identifiable executions points, such as method calls or object attribute assignments; (ii) pointcuts are composition queries that select a set of joinpoints from the space available of joinpoints; and (iii) advices are the new behaviour that extend, refine or replace the computation at selected joinpoints.

3 Empirical Settings

The stability assessment of PLAs involved in this empirical study is based on conventional metric suites for change impact [22] (Section 4) and modularity [24] (Section 5). These metrics were chosen as they have been worked as effective stability indicators in a number of previous studies (e.g. [9], [20], [24]). We also relied on the same change impact and modularity measures from the study of Figueiredo et al. [9], so that we could directly contrast their observations with our findings. Their study was only based on the comparison of conventional OO and AO implementations of MobileMedia (Section 3.1). They did not evaluate the complementary role of components and aspects on the design of stable SPLs. Then, we have replicated and extended their evaluation in order to compare the PLA stability of aspect-oriented COSMOS* implementations to the original implementations of MobileMedia.

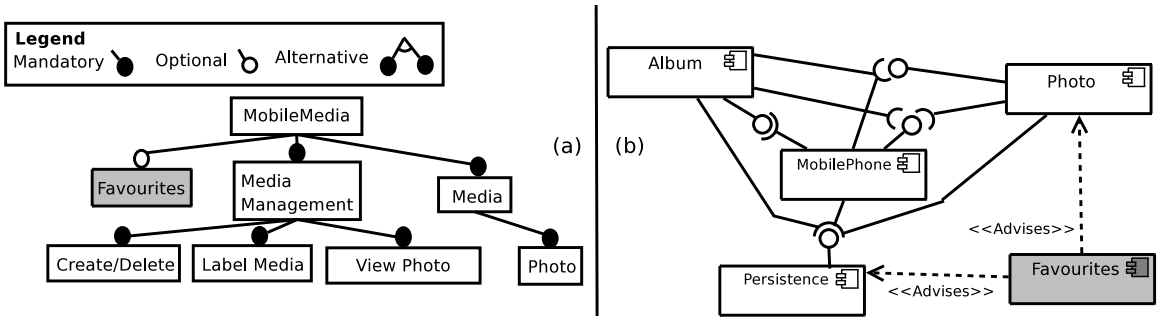

Figure 1: A feature model and COSMOS*-AO PLA model of MobileMedia SPL

We componentized eight releases of both versions of MobileMedia using COSMOS* implementation model. Hence, four implementations of MobileMedia were involved in this empirical study: (i) an OO version, (ii) an AO version, (iii) a COSMOS* version, and a (iv) COSMOS*-AO version. The first two versions were the original ones [9] and the last two versions were refactored from the two original ones, respectively.

3.1 Target software product line

In the following, in order to exemplify and evaluate our solution, we present a software application, called MobileMedia [9], which is a SPL for mobile applications that manipulates photo, music, and video on mobile devices, such as mobile phones. The system uses various technologies based on the Java ME platform, such as SMS, WMA and MMAPI. It has two versions with the same functionalities but implemented with different approaches: one uses AO programming and the other only OO programming. MobileMedia endured seven evolution scenarios, which led to eight releases. The scenarios comprises different types of changes involving mandatory, optional, and alternative features, as well as non-functional concerns. Both AO and OO versions of MobileMedia follow the same change scenario. The purpose of these changes is to exercise the implementation boundaries and, thus, assess the design stability of the PLA.

Figure 1(a) presents a partial view of the feature model of MobileMedia, following the notation proposed by Ferber et al. [8]. It is presented the features of the MobileMedia release four, such as `Create/Delete Photo`, `View Photo`, and `Label Photo`. In order to illustrate a evolution scenario, the optional feature `Favourites` is shown in gray, because it was added in release four.

Figure 1(b) shows a partial view of the component-based PLA with aspects of the release four. It is highlighted in gray the component `Favourites`, which was added in release four owing to the inclusion of the optional feature `Favourites`.

3.2 Case Study Definition and Execution

In order to execute the comparative study, we have performed the following steps:

- *Step 1.* Refactor eight releases (R1-R8) of the original OO version to COSMOS* version;
- *Step 2.* Refactor eight releases (R1-R8) of the original AO version to COSMOS*-AO version;
- *Step 3.* Collect change impact and modularity metrics for eight (R1-R8) COSMOS* releases;
- *Step 4.* Collect change impact and modularity metrics for eight (R1-R8) COSMOS*-AO releases;
- *Step 5.* Compare the results of COSMOS*-AO against COSMOS*, AO, and OO versions.

As a result of the refactoring Step 1 and Step 2, two different versions of MobileMedia SPL were generated, with eight releases each version. The size of the eight releases of the COSMOS* version is approximately 69 KLOC and 67 KLOC for the COSMOS*-AO version. During the execution of Step 1 and Step 2, we were strict in following the same implementation decisions made by the original MobileMedia developers, such as extracting exception handling code according to Castor et al. [10], and aspectizing all optional and alternative features. In addition, we have followed a number of well-known design practices to build modular designs[7]. During the execution of Step 3 and Step 4, we have used the same metrics suites of the original MobileMedia case study [9].

4 Change Impact Analysis

In this section, we discuss the change impact on components and on operations. The more resilient a PLA is, the minor is the change impact on its components and operations. The impact is measured by the number of components or operations added, changed, and removed in each release. The change impact is classified in two groups: components (Table 1(a)) and operations (Table 1(b)). Each group is divided in three subgroups: changed, removed, and added. Tables 1(a) and 1(b) show the impact in each release (R2 to R8) and the total impact of each version. The total number of removed operations and changed operations were minor on COSMOS*-AO PLA than on other PLAs. The total number of added components and added operations on COSMOS* PLA were slightly minor than on COSMOS*-AO PLA. In order to discuss these results from the feature point of view, we have classified the evolutionary changes into three groups: addition of optional features (R4, R5, and R6), addition of alternative features (R7 and R8), and addition of mandatory features (R2 and R3).

Addition of Optional Features. The overall results for these releases show that the COSMOS*-AO PLA required fewer changes and removals of components and operations than other PLAs. This COSMOS*-AO approach rely on pointcuts and advices (Section

Table 1: Change impact on components (a) and on operations (b)

(a)

			Releases							
			R.2	R.3	R.4	R.5	R.6	R.7	R.8	Total
Components	Changed	OO	5	8	5	8	6	12	22	**66**
		Cosmos*	4	7	4	7	6	10	14	**52**
		AO	5	10	2	8	5	16	9	**55**
		Cosmos*-AO	4	7	3	13	3	10	6	**46**
	Removed	OO	0	0	0	0	0	8	1	**9**
		Cosmos*	0	0	0	1	0	5	0	**6**
		AO	1	0	0	0	0	8	0	**9**
		Cosmos*-AO	0	0	0	0	0	4	0	**4**
	Added	OO	9	1	0	5	7	17	6	**45**
		Cosmos*	3	0	0	5	6	11	5	**30**
		AO	12	2	3	6	8	21	16	**68**
		Cosmos*-AO	2	2	2	4	5	17	8	**40**

(b)

			Releases							
			R.2	R.3	R.4	R.5	R.6	R.7	R.8	Total
Operations	Changed	OO	28	12	7	10	7	22	23	**109**
		Cosmos*	14	4	0	3	0	29	22	**72**
		AO	25	16	1	20	4	69	10	**145**
		Cosmos*-AO	10	6	0	3	0	0	0	**19**
	Removed	OO	0	2	0	19	0	71	13	**105**
		Cosmos*	0	0	0	40	0	45	1	**86**
		AO	2	2	0	20	0	63	13	**100**
		Cosmos*-AO	0	0	0	34	0	38	2	**74**
	Added	OO	32	21	3	36	37	110	45	**284**
		Cosmos*	11	12	0	42	49	84	39	**237**
		AO	49	28	10	37	47	118	71	**360**
		Cosmos*-AO	11	23	14	42	26	89	58	**263**

2) to modularly implement optional features creating new components, instead of generating scattered code in existent ones. For instance, in the addition of the optional feature Favourites, as shown in Figure 1(b), the main modification was the inclusion of the `Favourites` component on COSMOS*-AO PLA. In contrast to COSMOS*-AO PLA, this feature implementation was scattered in COSMOS* PLA design, thus increasing the change impact on its elements. However, creating components to implement optional features on COSMOS*-AO version increased the number of added components and operations more than other approaches. The results for addition of optional features highlights an interesting stability benefit of the joint use of aspects and components: the COSMOS*-AO PLA is more open to extensions and closed to modifications, conforming to the Open-Closed Principle [18].

Addition of Alternative Features. The overall measures were similar to those described in the addition of optional features, which means that the COSMOS*-AO PLA was the least impacted on the number of components and operations changed and removed. Once more, the COSMOS*-AO approach takes advantage of the obliviousness properties of aspects and the high cohesion of component-based approach to avoid change propagation on other architectural elements. For example, in the R8 of COSMOS*-AO PLA, we have created a cohesive component that supports and mediates the use of all alternative media (photo, music, and video) decreasing the impact on its architectural elements. It was not possible to create such component in other PLAs. In the COSMOS* and OO PLAs, the code necessary to combine the use of different media types was scattered in several components. In the AO PLA, it was required to create an aspect for each combination of media types. Similar to the addition of optional features, COSMOS*-AO PLA added more components and operations than COSMOS* PLA, because the COSMOS*-AO approach relies on pointcuts and advices to extend components creating new ones instead of changing them.

Addition of Mandatory Features. The overall results are similar for addition of mandatory features on COSMOS*-AO PLA and on COSMOS* PLA. COSMOS*-AO PLA has a slightly better measures in R2 than COSMOS* PLA. Nevertheless, COSMOS*-AO PLA has a slightly worse measures in R3 than COSMOS* PLA. The results are similar

because most of mandatory features in COSMOS*-AO were implemented using pure OO, which led to a code structure similar to the COSMOS* implementation.

5 Modularity Analysis

This section presents the results for the modularity analysis according to three metrics suites, namely separation of features (Section 5.1), coupling, and cohesion (Section 5.2). These metrics were chosen because they are previously-validated stability indicators as presented in several experimental studies(e.g. [9][10][12]).

The metrics for coupling and cohesion were defined based on conventional OO metric suites [6]. Furthermore, this metric suite includes the Feature Diffusion over Modules metric, in order to quantify the degree to which a single a feature is scattered on a system [22]. In the component-based PLAs, modules are considered as equivalent to components. The majority of these metrics can be automatically collected by applying metric tools, such as Aopmetrics [1] and AJATO [9].

5.1 Separation of Features

Figure 2 presents how three features, namely Favourites, Label Media, and Persistence, are scattered on the PLAs involved in this study. Figures 2(a) and (d) show that the optional feature Favourites is less scattered on COSMOS*-AO than on other versions thanks to the use of pointcuts and advices. Similar behaviour is also observed in other optional features implemented with aspects (e.g. Sorting), but due to space limitations they are not presented in this paper. This result indicates that the implementation of optional features combining components and aspects increase PLA modularity, which contributes to reduce the change impact caused by the addition of these features (see Section 4).

The scattering of Label Media (Figures 2 (b) and (e)) is similar on all PLAs because this feature was implemented in a similar way on all versions (i.e. not using aspects). Figures 2 (c) and (f) present the scattering of the Persistence feature. The COSMOS* implementation managed to separate the persistence feature better than COSMOS*-AO implementation. Our decision to implement optional and alternative features using aspects on COSMOS*-AO version harmed the modularity of some mandatory features. For instance, Photo, Music and Video alternative features depend on Persistence, once every photo, song or video must be persisted. Hence, as new optional and alternative features are included over the different releases, the number of components that contains mandatory features increases. These results are similar to those presented by Figueiredo et al.[9]

5.2 Cohesion and Coupling

Figure 3 shows the average lack of cohesion of methods (LCOM) from R1-R8. The COSMOS*-AO version has the lowest lack of cohesion. First, because COSMOS* model explicits the mapping between a component-based software architecture and source code, which maintains architectural modularity at the implementation level. Second, all COSMOS*-AO

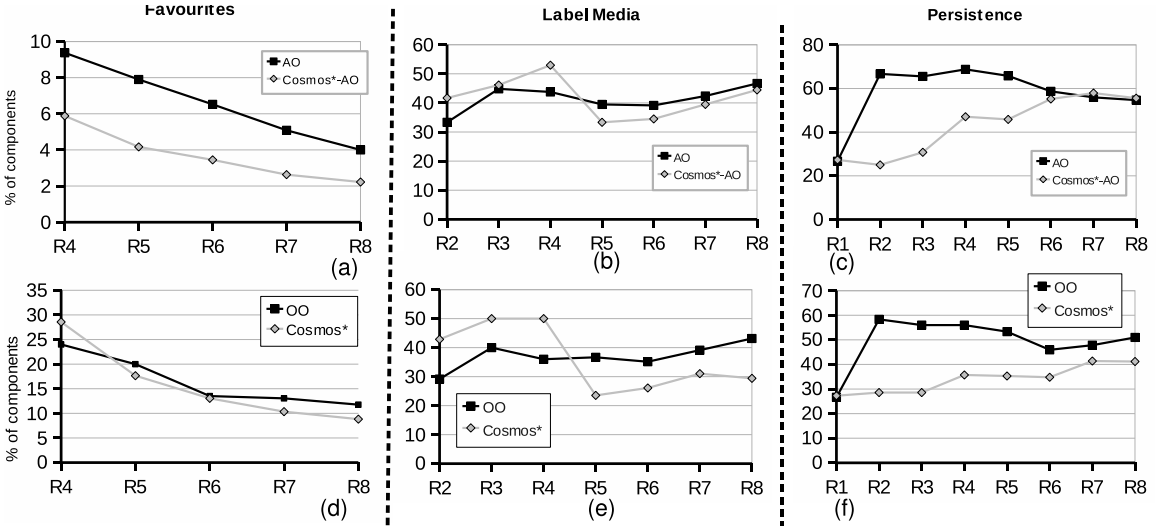

Figure 2: Feature Diffusion over Modules of all versions of MobileMedia

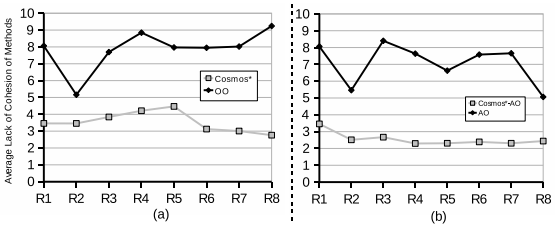

Figure 3: Average Lack of Cohesion of Methods

releases have a number of components greater than the number of components of COSMOS* releases. Then, there is a greater number of components implementing the same functionalities. Hence, each component of COSMOS*-AO PLA has a smaller scope than those of COSMOS* PLA, which increases its cohesion. Since high cohesion is a property of a modular architecture, the low lack of cohesion of the COSMOS*-AO PLA contributes to reduce the change impact on its architectural elements (see Section 4).

Figure 4 presents the average efferent coupling between components in each release, comparing original and refactored versions. COSMOS*-AO PLA has a higher coupling between its architectural elements than COSMOS* PLA. COSMOS*-AO components are usually more coupled than COSMOS* components, because COSMOS*-AO components must know the component they advise. For instance, in order to implement in a single component the Favourites feature, this component depends on two other components, Photo and Persistence (see Figure 1(b)). On the COSMOS* version, the Favourites feature is implemented in a scattered way by Photo and Persistence components, avoiding the creation of the Favourites component and its dependencies.

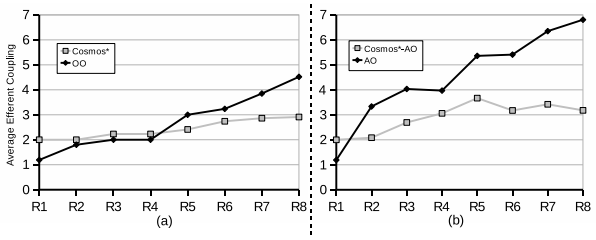

Figure 4: Average Efferent Coupling between Components

6 Related Work

The work of Figueiredo et al. [9] present a case study which assesses qualitatively and quantitatively the positive and negative impacts of AOP on design stability. Two SPLs were involved in their case study: MobileMedia (see Section 3.1) and BestLap. They conclude that AO versions of the SPLs tend to have more stable design, particularly when a change targets optional and alternative features.

Hoffman and Eugster [14] present a case study that aids in understanding the trade-offs between obliviousness and modularity. The exception handling mechanisms of three aspect-oriented applications were refactored using explicit join points (EJPs) in order to compare to OO and AO versions. They conclude that the greatest approach is a combination of the oblivious and explicit approaches.

Kvale et al[16] present a case study that investigates whether AO programming can help to build more easy-to-change COTS-based systems than OO. They compare how much effort is needed to: (i) integrate a COTS-based system; and (ii) replace COTS after integration. They conclude that fewer classes need to be changed when adding and replacing COTS using AO. They have evaluated the benefits of using AO to implement the glue-code between COTS and the system, while we have evaluated the using of AO in a SPL context.

7 Conclusions and future work

Evolutionary changes may increase maintenance costs of SPLs. A PLA is a key artefact to achieve a controlled evolution and, hence, it is important for organisations to understand how PLAs evolve and which approaches better support PLA stability. The main contribution of this paper is a novel analysis of the advantages and drawbacks of integrating components and aspects on design stable PLA. This approach was compared against the isolated use of AO, OO, and component-based approaches in the presence of heterogeneous changes. The overall results show that the COSMOS*-AO PLA is more resilient than the other PLAs. In particular, an interesting observation is that the synergistic combination of aspects and components seems to address the previously-observed configurability problems of aspect-oriented PLAs [9]. This finding is supported by the fact that the use of aspects and components (as supported by the COSMOS*-AO model) succeed in reducing the change propagation while adding multiple optional and alternative features in MobileMedia.

Threats to Validity and Future Work. One limitation of this study is that we focused on the evolution history assessment of one SPL. The same metric suites should be applied in the future to several and heterogeneous SPL applications in order to collect more reliable data. Nevertheless, there is a lack of empirical studies assessing state-of-the-art techniques for PLA stability on the literature. This study represents a significant step stone in this direction. Another limitation to be further addressed is the use of other aspect-aware design strategies for component-based PLAs. In fact, we are working on a new component model that extends COSMOS* implementation model to represent architectural elements with explicit interfaces representing architectural joinpoints. This new component model aims at providing the mapping between a component-based PLA with aspects and source code.

8 Acknowledgements

Leonardo P. Tizzei is supported by Capes/Brazil under grant 05866/2007. Marcelo Dias is supported by Fapesp/Brazil under grant 2008/02501-9. Cecília M. F. Rubira is partially supported by CNPQ/Brazil Productivity grant number 301446/2006-7.

References

[1] Aopmetrics. tigris.org. http://aopmetrics.tigris.org/.

[2] Corba component model specification. Technical Report formal/06-04-01 - Version 4, OMG, 2006.

[3] V. Alves, P. M. Jr., L. Cole, P. Borba, and G. Ramalho. Extracting and evolving mobile games product lines. In *LNCS*, volume 3714/2005, 2005.

[4] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wüst, and J. Zettel. *Component-based product line engineering with UML*. Addison-Wesley, Boston, MA, USA, 2002.

[5] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2003.

[6] S. Chidamber and C. Kemerer. A metrics suite for oo design. *IEEE TSE*, 20(6):476–493, 1994.

[7] P. Clements and L. Northrop. Software product lines: Practices and patterns. *Addison-Wesley*, 2002.

[8] S. Ferber, J. Haag, and J. Savolainen. Feature interaction and dependencies: Modeling features for reengineering a legacy product line. In *In Proc. of the Second Int. (SPLC), LNCS*, 2002.

[9] E. Figueiredo, N. Camacho, C. S. M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. Filho, and F. Dantas. Evolving software product lines with aspects: an empirical study on design stability. In *ICSE*, 2008.

[10] F. C. Filho, N. Cacho, E. Figueiredo, R. Maranhão, A. Garcia, and C. M. F. Rubira. Exceptions and aspects: the devil is in the details. In *SIGSOFT '06/FSE-14: Proc. of the 14th ACM SIGSOFT Int. symposium on FSE*, pages 152–162, NY, USA, 2006. ACM.

[11] R. E. Filman, T. Elrad, S. Clarke, and M. Aksit. *Aspect-Oriented Software Development*. Addison-Wesley Professional, October 2004.

[12] A. Garcia, C. Sant'anna, E. Figueiredo, U. Kulesza, C. Lucena, and A. von Staa. Modularizing design patterns with aspects: a quantitative study. In *Proc. of the 4th Int. Conf. on AOSD*, pages 3–14, NY, USA, 2005. ACM Press.

[13] L. A. Gayard, C. M. Rubira, and P. A. Guerra. COSMOS*: a COmponent System MOdel for Software Architectures. Technical Report IC-08-04, Institute of Computing, University of Campinas, February 2008. In English, 58 pages.

[14] K. Hoffman and P. Eugster. Towards reusable components with aspects: an empirical study on modularity and obliviousness. In *ICSE*, pages 91–100, New York, NY, USA, 2008. ACM.

[15] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. marc Loingtier, and J. Irwin. Aspect-oriented programming. In *Proc. of ECOOP*, pages 220–242. Springer-Verlag, 1997.

[16] A. A. Kvale, J. Li, and R. Conradi. A case study on building cots-based system using aspect-oriented programming. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1491–1498, NY, USA, 2005. ACM.

[17] M. M. Lehman, J. F. Ramil, and D. E. Perry. On evidence supporting the feast hypothesis and the laws of software evolution. In *METRICS*, page 84, Washington, DC, USA, 1998. IEEE Computer Society.

[18] B. Meyer. *Object-oriented software construction (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[19] P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz. An empirical study of software reuse vs. defect-density and stability. In *ICSE*, pages 282–292, Washington, DC, USA, 2004. IEEE Computer Society.

[20] C. Nunes, U. Kulesza, C. Sant'Anna, I. Nunes, A. Garcia, and C. Lucena. Comparing stability of implementation techniques for multi-agent system product lines. In *Proc. 13rd European CSMR*, Kaiserslautern, Germany, March 2009.

[21] J. Oldevik. Can aspects model product lines? In *EA '08: Proc. of the 2008 AOSD workshop on Early aspects*, pages 1–8, New York, NY, USA, 2008. ACM.

[22] C. Sant'anna, A. Garcia, C. Chavez, C. Lucena, and A. v. von Staa. On the reuse and maintenance of aspect-oriented software: An assessment framework. In *Proc. XVII Brazilian Symposium on Software Engineering*, 2003.

[23] C. Szyperski. *Component Software*. Addison-Wesley, 2002.

[24] S. Yau and J. Collofello. Design stability measures for software maintenance. *IEEE TSE*, 11(9):849–856, 1985.