AN ASPECT-ORIENTED APPROACH FOR THE DESIGN OF PRODUCT LINE ARCHITECTURES

Cortés Verdín Karen [1]
Lemus Olalde Cuauhtémoc [2]
van der Hoek André [3]
Fernández Peña Juan Manuel[1]

[1]kcortes@uv.mx, jfernandez@uv.mx
[2]clemola@cimat.mx
[3]andre@ics.uci.edu

RESUMEN

Un enfoque de Líneas de Productos de Software (LPS) mejora la productividad, reduce su tiempo de mercadeo y obtiene productos de calidad. El Desarrollo de Software Orientado a Aspectos (DSOA o AOSD en inglés) busca una adecuada separación de intereses para obtener software fácil de mantener, evolucionar, reutilizar, personalizar y entender. Aquí se presenta AOPLA (Aspect-Oriented Product Line Architecture). AOPLA es un método de diseño de arquitecturas de LPS con un enfoque orientado a intereses. AOPLA permite una identificación temprana de intereses y su tratamiento a lo largo del proceso, de tal manera que éstos son resueltos en la arquitectura.

ABSTRACT

A Software Product Line (SPL) approach improves productivity, reduces time to market and achieves high quality products. Aspect-Oriented Software Development (AOSD) seeks proper separation of concerns in order to obtain software that is easy to maintain, evolve, reuse, customize and comprehend. This paper presents AOPLA(Aspect-Oriented Product Line Architecture). AOPLA is a product line architecture (PLA) design approach with a concern-oriented focus, which allows for an early identification of concerns and their handling along the process in such a way that concerns are addressed during architecture modeling.

Keywords: Software Product Lines, Software Architecture, Aspect-Oriented, Concern-Oriented, Multi-Dimensional Separation of Concerns (MDSOC), Quality model

I. INTRODUCTION

A Software Product Line (SPL) is a "set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"[2]. The main characteristics of a SPL approach are: 1) architecture-

[1] School of Statistics and Informatics, Universidad Veracruzana, Mexico
[2] Center for Research in Mathematics (CIMAT, A.C.), Mexico
[3] University of Carliforma, Irvine, USA

centric development, 2) two-tier development organization and, 3) proactive and planned reuse of core assets. Core assets form the basis for a SPL. A core asset is an artifact of software development that is to be reused in the development of the products of the SPL. One of such core assets is the software architecture or product line architecture (PLA). A software architecture is defined as " . . . the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them" [3]. As such, a software architecture not only defines the functionality of the system but also its quality attributes. For a SPL, the PLA is even more important since it:

1. supports the commonality and variability of the products within the PL,
2. exhibits the PL-specific quality attributes and those quality attributes that pertain to applications and
3. is general enough to allow for the derivation of PL products.

The PL-specific quality attributes correspond to those defined in CAFÉ project [4]. Such attributes are:

1. Variability. The capability of an asset to contain common and varying parts thereby covering aspects of different product line members.
2. Derivability. Capability to derive a concrete, product-specific asset from a generic core asset.
3. Reusability. Capability to reuse and existing asset for different product line members.
4. Rateability. Capability to estimate a core asset's worth.
5. Integrability. Capability to integrate a system-specific asset into the PL infrastructure.
6. Correctness. Extent to which an asset satisfies its specification and fulfils the PL's mission objectives.
7. Evolvability. Capability of an asset to evolve over time thereby dealing with growing complexity and demand as well as continuous change.
8. Manageability. Capability to manage core assets and asset configurations.
9. Maintainability. Capability of an asset to be modified.

Aspect-Oriented Software Development (AOSD) approaches seek separation of concerns in order to obtain software that is easy to maintain, evolve, reuse, customize and comprehend. AOSD provides the means for the identification, modularization, representation and composition of crosscutting concerns [5]. Crosscutting concerns or aspects are "such quality factors or functionalities of software that cannot be effectively modularized using existing software development techniques" [5]. Separation of concerns can be applied from the very beginning of the software development process (i.e., requirements engineering and architecture design phases). A concern in these early phases can crosscut an artifact of the requirements or architecture design phases. These kinds of concerns are known as Early Aspects (EA) [6] [7]. During architecture design it is important that aspects are properly identified and modularized, otherwise, they can lead to tangled code during implementation. This, in turn, would compromise the maintainability, integrability, manageability and evolvability of the software system.

AOPLA (Aspect-Oriented Product Line Architecture) was envisioned as an architecture design approach with an early identification and handling of concerns. Such an early identification and handling of concerns is achieved with a concern-oriented focus

starting during the domain engineering phase. In this way, concerns are treated as first-class entities from the beginning. In this way, AOPLA allows for obtaining a generic and evolvable PLA. Such a PLA supports commonality and variability, and complies with the PL-specific quality attributes and the products specific quality attributes.

This document presents AOPLA and is organized in the following way. The second section describes the AOPLA process. The third section corresponds to the description of the case study: Ancora-Soft and the AOPLA process followed for the case study. Ancora-Soft is a case tool that implements ANCORA[8]. Section number four corresponds to the evaluation of the approach. Finally, conclusions are given.

II. AOPLA PROCESS

The process followed in AOPLA is shown in Figure 1 on the next page. The description of each phase is described below.

Business case development. The business case is a means of making a decision on whether to pursue a new business opportunity or not. In addition, during the development of the business case the PL portfolio is defined. This portfolio is an artifact or core asset for AOPLA. The portfolio consists on a list of products and their corresponding capabilities and requirements. Is in this portfolio where the first opportunities of reuse can be identified.

Domain engineering. The focus of this phase is on the analysis and modeling of applications within the domain in order to find opportunities of reuse. The activities performed are based on FODA [9]. It is during this phase that an early identification and modeling of concerns is performed. This is done by using Cosmos [7]. Cosmos is a concern space modeling schema that allows for representing concerns as first-class entities. Since concerns are represented as first-class entities the tyranny of the dominant decomposition is avoided. In this way, a multi-dimensional separation of concerns is achieved (MDSOC). The concerns identified and modeled in Cosmos are properly handled during the rest of the AOPLA process assuring their handling during the architecture modeling phase.

Architecture modeling. The first two activities in this phase correspond to use case modeling and requirements specification. During use case modeling variability and commonality is expressed in the use case model and use cases descriptions. With all the knowledge acquired during domain engineering, quality attributes are easily identified and specified during requirements specification. These requirements will be considered later during the development of the product line quality model (PLQM). It must be noted that such quality attributes correspond to the PL-specific quality attributes and also to the quality attributes that apply to applications within the domain.

The next activity in architecture modeling corresponds to CORE (Concern-Oriented Requirements Engineering) [10]. CORE performs a projection or composition of concerns (the concerns previously identified and modeled in Cosmos) in order to identify and handle conflicts among concerns. The main output of CORE is the determination of the mapping and the influence of concerns on later development stages. The mapping of concern determines whether the concern will be resolved as a component, an aspect or an architectural decision. The identified architectural decisions

are considered during the development of the PLQM while the identified components and aspects are considered during the development of architectural views.

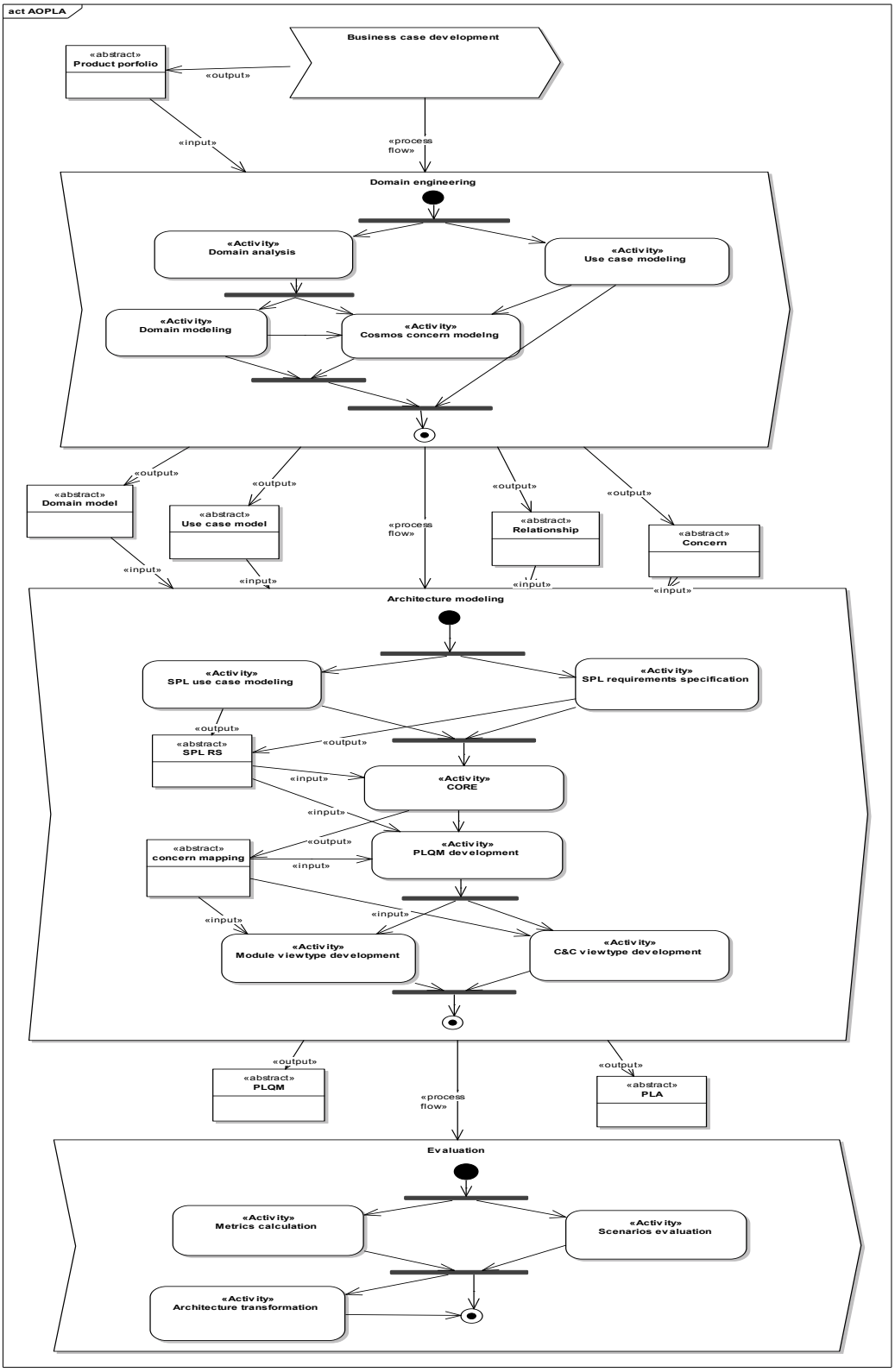

FIGURE 1. AOPLA PROCESS.

After CORE, the product-line quality model (PLQM) is developed . This model encompasses the specification of quality attributes metrics and scenarios, the selection of means and patterns for such quality attributes and the selection of the architecture's *viewtypes* and views.

The development of the Module and Component&Connector (C&C) *viewtypes* is the actual architecture modeling. During this modeling the selected views are developed considering the architectural decisions identified during CORE and in the PLQM.

The last activity of architecture modeling is the PLA evaluation. For this, the metrics and scenarios specified in the PLQM are used. If necessary, transformations are done on the PLA in order to achieve the quality attributes.

III. ANCORA-SOFT CASE STUDY

The case study selected for the evaluation of AOPLA was Ancora-Soft. Ancora-Soft is a case tool supporting ANCORA [8]. ANCORA is a requirements analysis methodology which has been widely applied. ANCORA has been used in requirements analysis of systems between 100 and 600 function points. ANCORA provides the analyst with several tools that permit adequate stakeholders' involvement in the process. The main modeling tool of ANCORA is the script. Like in a theater play, the cast (users) will play several roles. With the requirements analyst's help, the script is written by the stakeholders. Other main feature is that the methodology promotes reuse. ANCORA encompasses several ways of storing elements from previous systems so that they can be used in the development of new systems. Other functionality is included in Ancora-Soft, for example: function points calculation, use case points calculation, tests (cases, plans and procedures) and risk management.

The phases and activities described above were developed for Ancora-Soft PLA. During the business case phase the product portfolio was defined. This product portfolio contains a total of four products and their corresponding capabilities and requirements. The four products are:

1. Product 1. In addition to being used for requirements analysis and software development, ANCORA is used for teaching the fundamentals of Software Engineering. Therefore, the intended users of Product 1 are Software Engineering students at the B. Sc. degree level. The capabilities and requirements included in this product correspond to the basic elements of ANCORA ands those related to function points calculation, user manual generation, semantic objects modeling and entity-relationship modeling.

2. Product 2. The intended users of this product are software developers. This product includes the capabilities and requirements of Product 1 plus use case point calculation, test support (elaboration of test cases, plans and procedures), defects registration and follow-up, technical reviews and the generation of the software requirements document.

3. Product 3. This product is aimed at software development teams using ANCORA for requirements analysis and the connections of the methodology to software development approaches such as the Unified Process. Product 3

therefore, encompasses the capabilities and requirements from products 1 and 2 and also those that correspond to the connection to other methodologies, the support for collaboration among developers teams and the connection to other CASE tools by using XML files.

4. Trial version. This product corresponds to a reduced version of Ancora-Soft. The objective of this product is to let prospective users and customers have a first contact with the methodology and the CASE tool.

The artifacts developed during domain engineering correspond to context, functional, feature entity-relationship and Cosmos models. Cosmos schema considers logical and physical concerns, relationships and predicates [7]. Logical concerns are those concepts or matters of interest in relation to a software system or an artifact. Among logical concerns properties and topics are important to mention. Properties are logical concerns that characterize other logical concerns, while topics are theme-related concerns. Topics are a way of identifying crosscutting concerns.

During Cosmos, the logical concerns identified and modeled corresponded to entities, functionality and features. Properties corresponded to PL-specific quality attributes and quality attributes of applications in the domain. The choice of PL-specific quality attributes for AOPLA were: evolvability, reusability and derivability. Generality was also considered, since a PL must be general in order to enable the derivation of products. The quality attributes of the applications were modifiability, security and portability were considered. Most of the logical concerns are identified as topics, specially the properties, because they are constantly repeated (or have a crosscutting relationship) within the model. For relationships, *significant-for relationships* makes explicit the relationships between functionality, features and entities. Table 1 shows an excerpt of the *significant-for* relationship between functionality and features. Another important relationship is *applies-to*. This relationship makes explicit the relationship of properties with concerns and it also helps to appreciate the crosscutting relationship that a quality attribute has with the corresponding concerns.

TABLE 1. *SIGNIFICANT-FOR* RELATIONSHIP (FUNCTIONALITY IS SIGNIFICANT FOR FEATURE).

Functionality	Feature
Cost.Function Points calculation	Function Points
Cost. Use Case Point calculation	Use Case Points
NSN.Create	NSN
NSN.Import	NSN
NSN.Export	NSN
Glossary.Create	Glossaries and tables.Create
Table.Create	
Glossary.Update	Glossaries and tables.Edit
Table.Update	
Script.Generate development log	Script management. Development Log generation
Dialogue.Create	Script management.Dialogue
Script.Create	Script management.Create script
Track.Create	Script management.Track
Development log.Create	Development log.Create
Development log.Generate	
Development log.Update	Development log.Update
Development log.View	Development log.View

In use case modeling, extend relationships were used to model variability and commonality. Quality attributes were specified during requirements specification. The specification included the PL-specific quality attributes and the quality attributes of the products.

During CORE concerns in Cosmos model are considered. CORE encompasses a concern composition step. This composition is done in order to: 1) determine and handle conflicts among concerns, 2) establish the dimensions of concerns. A dimension of concern consists of influence of the concern for the rest of the development process and of mapping (component, design decision or aspect) of the concern. This composition is done for every product of the PL. Table 2 below shows dimensions of concerns for the trial version.

TABLE 2. INFLUENCE AND MAPPING OF CONCERNS FOR THE TRIAL VERSION PRODUCT

Concern	Influence	Mapping
Function Points	Spec, Design, Impl	Component
NSN	Spec, Design, Impl	Component
Glossaries and tables	Spec, Design, Impl	Component
Development log	Spec, Design, Impl	Component
Script management	Spec, Design, Impl	Component
Technical reviews	Spec, Design, Impl	Component
Test cases	Spec, Design, Impl	Component
Defects	Spec, Design, Impl	Component
Risk management	Spec, Design, Impl	Component
MS-Office Compatibility	Spec, Design, Impl	Component
Printing	Spec, Design, Impl	Component
Project	Spec, Design, Impl	Component
Portability	Architecture, Design, Impl	Decision
Export	Spec, Design, Impl	Aspect
Data dictionary	Spec, Design, Impl	Component

Spec=Specification, Imp=Implementation

The quality attributes considered in the PLQM were previously analyzed in detail during CORE. Therefore, the definition of the product line quality model (PLQM) is enhanced because there is a better understanding of the domain, of the quality requirements and, above all, the conflicts among them (if any) have already been resolved.

The PLQM supported also the selection of *viewtypes* and views for the PLA. The basic set of views is based on the SEI's "Views & Beyond" approach[10]. For AOPLA the set of views consists of the Module and Component&Connector (C&C) *viewtypes* [11] . The selection of views is shown in Table 3 on the next page.

The ADR(Abstract Data Repository) ABAS (Attribute-Based Architectural Style) [12] was selected. The ADR addresses modifiability and evolution, since it provides for the modification of consumers and producers as well as the inclusion of new consumers and producers. This characteristic of the ABAS improves the derivation of specific products and the inclusion of new products in the PL. From the styles of the C&C viewtype[11], no specific style was selected to address a quality attribute. Instead, the ADR was further detailed during C&C *viewtype* development.

During the *viewtypes* development, the patterns selected during the development of the PLQM are used. During this process, tactics, principles and techniques specified in the PLQM are also applied. In parallel, CORE mappings are incorporated in the PLA. Architectural decisions have already been considered during the development of the PLQM. The remaining mappings are those that correspond to components and aspects.
In order to continue addressing MDSOC it is necessary that during architecture modeling, the relationships of concerns and the portfolio definition are updated. The product derivation process consists mainly in the development of each product C&C view. Furthermore, the ADR was then translated into the C&C *viewtype* for each one of the products. Figure 2 on the next page shows the C&C view for Risk consumer and producer. The model encompasses a view for each one of the products. Within the C&C *viewtype* an aspectual view is included. This view is included to fully describe aspects and their relationships. The aspectual view contains all components of the PLA showing the relationships of aspectual components (aspectual components are shown with an <<aspect>> stereotype).

TABLE 3. SELECTION OF VIEWS ACCORDING TO QUALITY ATTRIBUTES.

Quality attribute	View	Viewtype
Reusability, derivability, generality	Generalization, uses style	Module
Modifiability	Decomposition, layers	Module
Evolvability	Decomposition layers, Generalization	Module
Portability	Layers	Module
Security	Deployment	Allocation

IV. EVALUATION

The evaluation of AOPLA was done in the following phases:

AOPLA evaluation. The PLQM scenarios evaluation for AOPLA probed that the PLA fulfilled the quality attributes. This, in addition to the metrics evaluation, gave as result that no transformation of the PLA was needed.

ATAM evaluation. The goal of ATAM (Architecture-Tradeoff Analysis Method) [13] is to understand the consequences of architectural decisions with respect to the quality attribute requirements of the system. ATAM is meant to be a means of detecting areas of potential risk within the architecture. The conclusions of this evaluation were that Ancora.-Soft's AOPLA fulfilled the required quality attributes of: evolvability, modifiability, portability, generality, derivability and reusability. Also, the architectural approaches, techniques, tactics and principles considered in the architecture proved to

support all the expected scenarios, including those brainstormed by the stakeholders in one of ATAM's steps.

COSAAM evaluation. COSAAM (Concern-Oriented Architecture Analysis Method) [14] is a scenario-based method for the analysis of aspect-oriented architectures. Its aim is to explicitly identify and specify architectural aspects and make them transparent early in the software development life cycle. COSAAM also includes identification of concerns, dependency analysis of architectural modules and the measurement of scattering and tangling. The conclusions from COSAAM were that:

1. AOPLA process adequately handles concerns and incorporates them into the architecture
2. AOPLA maps concerns into modules that are highly cohesive, makes and early detection of inherently crosscutting concerns addressing them as aspects or architectural decisions.
3. Accidentally crosscutting concerns are natural due to activities performed and artifacts generated during the domain analysis phase. These concerns are characterized as scattered (whether direct or indirect).

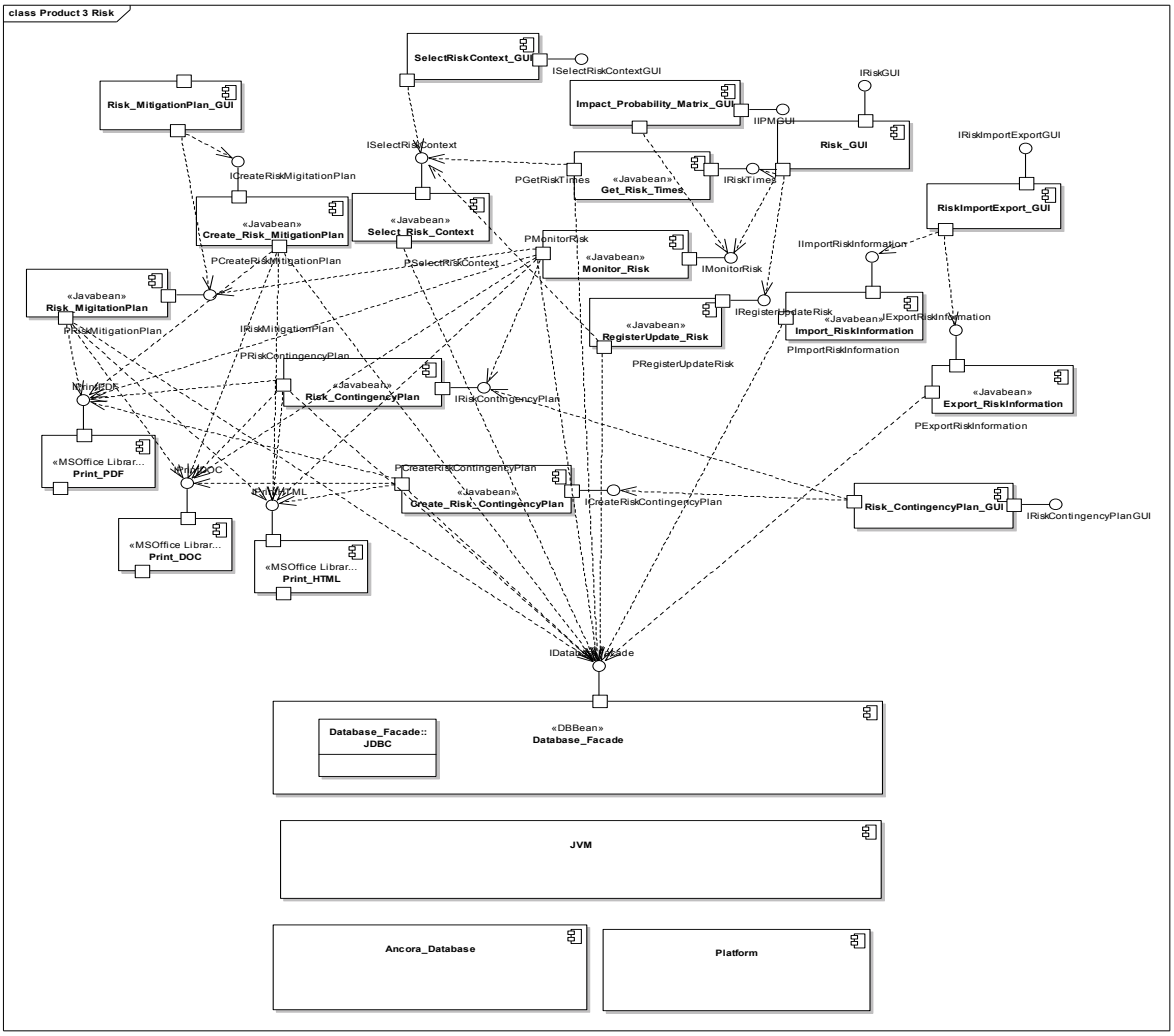

FIGURE 2. RISK CONSUMER AND PRODUCER C&C VIEW.

V. CONCLUSIONS

AOPLA is an Aspect-Oriented Product Line Architecture design approach that supports multi-dimensional separation of concerns (MDSOC). Cosmos is integrated into AOPLA to achieve an early identification of concerns. Such an early identification of concerns enables the identification of aspects, the selection of architectural approaches and the handling of concerns during the rest of the process. During requirements engineering, CORE (Concern-Oriented Requirements Engineering) enables the identification and handling of conflicts among concerns and also establishes the mappings and influence of concerns. These mappings and influence are used during architecture modeling in the development of views.

Since the product line architecture defines the quality of the PL products, it was important to incorporate a quality model into AOPLA. In order to this, the CAFÉ quality model was used. The CAFÉ quality model encompasses the definition of quality attributes and their corresponding metrics, scenarios, principles, tactics and techniques. This model was later used for the evaluation of the case study PLA and also for the evaluation of the approach.

The evaluation of ATAM concluded that the architectural decisions made in AOPLA actually fulfilled the specified quality attributes. As result of COSAAM, it was concluded that AOPLA obtains highly cohesive modules and the scattering of concerns is very low.

PLA design being the matter of interest, the modeling of commonality and variability is important. For AOPLA this is accomplished using a feature model during domain engineering and UML extensions (as proposed in [15]) during the development of the architectural views. One of the areas of future work is precisely the incorporation of a more robust variability model. Another area is the development of an aspect-oriented architectural UML profile for the description of AOPLA. Architecture description is important in order to fully exploit the architecture. Without an adequate architecture description, the architectural knowledge cannot be transmitted and, therefore, will not be reflected in the derived products.

REFERENCES

[1] Paul Clements, Linda Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, 2001.

[2] Len Bass, Paul Clements, and Rick Kazman, Software Architecture in Practice, Addison-Wesley, USA, 2003

[3] http://www.esi.es/en/Projects/Cafe/cafe.html (accessed 2005)

[4] Ruzanna Chitchyan, Awais Rashid, Peter Sawyer., Alessandro Garcia, Mónica Pinto Alarcón, Jethro Bakker , Bedir Tekinerdogan, Siobhán Clarke, and Andrew Jackson, Survey of Aspect-Oriented Analysis and Design Approaches (AOSD-Europe-ULANC-9). Editor(s): R. Chitchyan, A. Rashid. Lancaster, UK, University of Lancaster, 2005.

[5] Elisa Baniassad, Paul C. Clements, Joao Araujo, Ana Moreira, Awais Rashid, and Bedir Tekinerdogan, "Discovering Early Aspects", IEEE Software, vol.23, no. 1, January/February 2006, pp. 61-70.

[6] http://www.early-aspects.net/ (accessed 2006).

[7] Stanley M. Sutton Jr, Isabelle Rouvellou, "Modeling of Software Concerns in Cosmos", Proceedings of the 1st International Conference on Aspect-oriented software development, Enschede, The Netherlands, 2002.

[8] A. Sumano López, “Método para el análisis de requerimientos de software con un enfoque psicológico, social y lingüístico conducente al reuso”, PhD Dissertation, IPN, México, 2002.
[9] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.
[10] Ana Moreira, Joao Araujo, and Awais Rashid, "A Concern-Oriented Requirements Engineering Model", Proceedings of the Int'l Conference on Advanced Information Systems Engineering (CAiSE) 2005, Porto, Portugal, 2005.
[11] Paul Clements, Felix Bachman, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, Documenting Software Architectures: Views and Beyond, Addison-Wesley, USA, 2003.
[12] Mark Klein and Rick Kazman , Attribute-Based Architectural Styles (CMU/SEI-99-TR-022). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999
[13] Rick Kazman, Mark Klein, Paul Clements, ATAM: Method for Architecture Evaluation (CMU/SEI-2000-TR-004. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
[14] Frank Scholten, “Concern-Oriented Architecture Analysis Method”, MSc Thesis, University of Twente, Dept. of Computer Science, Software Engineering Group, The Netherlands, February 2007.
[15] Hassan Gomaa, Designing Software Product Lines with UML, Addison-Wesley, 2005.

María Karen Cortés Verdín

Profesor de Tiempo Completo de la Facultad de Estadística e Informática de la Universidad Veracruzana. Licenciada en Informática de la Universidad Veracruzana en 1989. Obtuvo el grado de Master of Science in Information Systems Engineering en 1995 del Instituto de Ciencia y Tecnología de la Universidad de Manchester (UMIST), UK. Grado de Maestría en Ingeniería de Software en 2005 del Centro de Investigación en Matemáticas A.C. (CIMAT); México y Doctorado en Ciencias de la Computación en 2009 del Centro de Investigación en Matemáticas A.C. (CIMAT), México.