Design Aspects for Describing Frameworks

Federico Balaguer
Software Architecture Group –
Computer Science Department
Univ. of Illinois at Urbana-Champaign
Urbana, Illinois - 61802
(217) 333-1043
balaguer@uiuc.edu

ABSTRACT
The poster presents an extension to UML for describing design aspects of frameworks. Aspects are documented by applying a UML Profile called "Framework Description" to class diagrams. Design Aspects of frameworks are useful for reasoning about extensions and instantiations scenarios as well as designing the application that relies on them.

Keywords
Framework Design, UML, UML-F, Framework Implementation, Design Aspects, Separation of Concern.

1. INTRODUCTION
Modern object-oriented systems are usually made from several frameworks. Each framework addresses a different part of the system, such as security, persistence, transaction management, or user interface. The system provides the glue that coordinates the components and customizes the frameworks. Thus, developers spend a lot of time instantiating and configuring frameworks.

Different types of frameworks require different instantiation techniques. Black-box frameworks are instantiated by specifying parameters that will produce the desired behavior. White-box frameworks, on the other hand, are instantiated by adding code to an existing class structure [3]. Most applications developed from frameworks use both techniques. Extending a framework is a different scenario, developers need to acquire a profound knowledge of the design and its domain in order to add new classes and methods.

The extension introduced usually follows the abstractions and principles that the framework mandates. Note that after the instantiation step developers are still one step away from building a real application where the instantiated frameworks are applied. Developers have to add the code that represent the application behavior and create objects from the frameworks.

Although, UML-F has been used for describing variation points within a framework [2], the approach does not apply well to black-box framework where the instantiation of the framework is based on parameterizing instances of classes provided by the framework [5][1]. Moreover, UML-F only considers one of the possible dimensions of concern [6], the instantiation face.

The poster presents an extension to UML (as a UML Profile) for describing aspects of a framework design. Each aspect captures variation points, domain invariants and other qualifiers of the framework design. Aspects systematically capture design decisions relevant for extending, instantiating and initializing frameworks.

2. DESIGN ASPECTS OF FRAMEWORKS
UML-F [2] is a UML extension for describing the information required for framework development and instantiation. UML-F introduces a set of fix tagged-value describing variation points and instantiation constraints. The elements introduced can be applied to different parts of a model. For example, {appl-class} is a tagged-value applicable to classes that are present after the framework is instantiated. Other tagged-values include, {variable} applicable to methods, {extensible} applicable to classes and {incomplete} applicable to generalization/realization relationships, {dynamic} is applicable to Extensible Interface, Classes and Methods when run-time instantiation is required.

UML-F is well suited for describing variation points at the instantiation face of a white-box framework. The inclusion of the {dynamic} tag does not provide enough expressive power to UML to describe instantiation of black-box frameworks. Instantiation of black-box frameworks depends on scripting languages or tools. Either way users (in this case the developers) don't need to know about the design of the framework but the abstractions (offered by the framework) and the legal relationships among them.

Figure **1** shows a simplified version of the design of PPL a persistence framework for Smalltalk and Java. The class diagram includes core classes of the framework such as PersistentObject, SessionManager, DatabaseConnection, AttributeSpec, TableSpec, and the RDBType hierarchy. Developers instantiating the framework have to make subclasses of PersistentObject and implement few operations such as: attributeSpec() and oidPrefix().

On the other hand, developers extending the framework for including new data types such URL, should create a new subclass on the RDBtype hierarchy with the required methods for mapping the type to the database. These two scenarios are examples of concerns traversing the framework: instantiating and extending respectively.

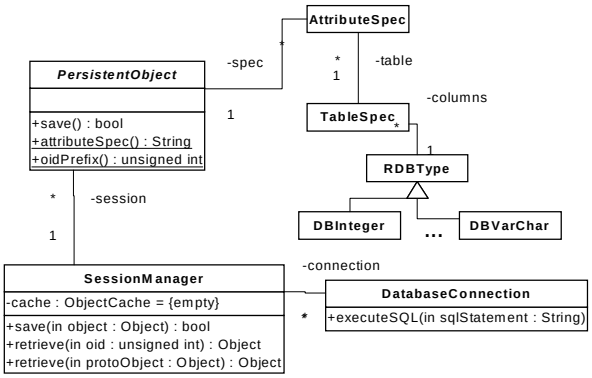

Figure 1 Persistence Framework

3. DESIGN ASPECTS OF FRAMEWORKS WITH UML

A design aspect captures variation points, invariants and other qualifiers of a framework design. Each aspect is specifically related with one of the dimension (instantiation, extension, configuration/initialization) in which a framework can be examined. Design aspects are different from programming aspects [4] in that design aspects capture the relationship between elements of a framework design (classes, operations, attributes, relations) and concerns within the three mayor dimensions mentioned before. Programming aspects allow managing variations in the code and flow of control of programs. Each aspect defines an implementation (between many alternatives) of a given concern.

The extension proposed in the poster is defined as a UML Profile called "Framework Description". The profile introduces stereotypes (such as: Aspect and crosscutting-point), a set of tagged-values and constraints that complement those introduced by UML-F. These new tagged-values describe attributes of the framework that are not variation points. For example, {final} is a boolean tag applicable to classes and methods, (cfg-constrain) is an OCL (Object Constrain Language) expression that captures invariants at the domain level and {empty} is a boolean tag applicable to methods meaning that subclasses has to implement it. Constrains introduced by the extension (cfg-constrain) refer to incompatibilities among the extended set of tagged-values, for example a given method within a particular aspect cannot be labeled as {extensible=true} and {final=true} or {incomplete=true} and {extensible=false} (which was legal in UML-F).

Design aspects do not add new elements to an existing model as it is proposed in [6] but they qualify existing elements of the framework. The approach does not assume any development process since it tries to capture the rational behind the design instead of producing it.

We are building a tool that manages aspects of a UML model. The tool is a wrapper over Visio 2000, it is implemented in Dolphin Smalltalk and based on COM technology.

Different frameworks are being documented based on these ideas such as: a framework for developing digital-libraries, the Observation framework along with a persistent framework. These frameworks can be downloaded from: http://www.uiuc.edu/ph/www/balaguer.

4. CONCLUSIONS

UML is not sufficient for expressing different concern dimensions of a framework, but it can be extended to be sufficient. This poster presents an on-going research extending UML with a profile for capturing design aspects of frameworks

5. ACKNOWLEDGMENTS

The research project is sponsored by Systems Integration Technology Center, Toshiba Corporation and University of Illinois at Urbana-Champaign.

6. REFERENCES

[1] D. D'Souza, A. Cameron Wills. 'Objects, Components and Frameworks with UML. The Catalysis Approach'. Addison-Wesley. 1999

[2] M. Fontoura, W. Pree, B. Rumpe. 'UML-F: A Modeling Language for Object-Oriented Frameworks'. Proceedings of ECOOP 2000. Springer Verlag. 2000

[3] R. Johnson, B. Foot. 'Designing Reusable Classes'. Journal of Object-Oriented Programming, 1(2):22-25, June/July 1988.

[4] G. Kickzales et. al. 'Aspect Oriented Programming' Proceedings of ECOOP 1997. Springer-Verlang. 1997

[5] M. Markiewicz, C. de Lucena. 'Object Oriented Framework Development'. Crossroads. Issue 7.4, Summer 2001. ACM

[6] P. Tarr et al. 'N Degrees of Separation: Multi-Dimensional Separation of Concerns'. Proceedings of ICSE 1999. ACM. 1999