ENHANCING FRAMEWORK DESIGN AND UTILIZATION

Marcus Felipe M. C. da Fontoura
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de S. Vicente, 225
Rio de Janeiro 22453-900
E-mail: mafe@inf.puc-rio.br
Advisor: Carlos José P. de Lucena

ACM'99 STUDENT RESEARCH CONTEST (GRADUATE)

ABSTRACT

This work proposes a new approach for framework design and utilization. We believe this approach will lead to a more explicit definition of the collaboration between the framework elements and will help the designer to better implement the framework variability through the use of patterns. Regarding framework utilization, we propose a different instantiation process, where a particular application is described by domain specific languages (DSLs) that are designed precisely for the framework domain. A major topic in our work is the semi-automatic derivation of the DSLs from the framework design structure. The use of formal methods will allow us to prove that the applications that can be derived from a framework through the "standard" instantiation process are exactly the same that can be specified as DSLs programs.

1. PROBLEM AND MOTIVATION

There are various application areas that are not established yet and for which new ideas and models are presently under development and evaluation. These are domains for which the viability of rapidly building new applications is essential and strategic from a practical point of view. Examples of application domains that can be classified in this category include Web-based Education, electronic commerce, biology, and financial market.

An interesting strategy for developing new applications for these domains is the development of object-oriented application frameworks [11, 9]. Frameworks can reduce the costs of developing applications since it allows designers and implementers to reuse their previous experience with problem solving at the design and code levels. Prior research has shown that high levels of software reuse can be achieved through the use of object-oriented frameworks. An object-oriented framework captures the common aspects of a family of applications, which can be seen as a domain theory [18].

Although object-oriented software development has experienced the benefits of using frameworks, a thorough understanding of how to identify, design, implement, use, and change them to meet evolving requirement needs is still object of research [18, 23].

Therefore, framework development is very expensive not only because of the intrinsic difficulty related to capturing the domain theory but also because of the lack of appropriate methods and techniques to support framework specification and design.

Framework utilization is also a problem noticed in companies that use this technology to create applications [http://www.objectime.com/OOPSLA98/frameworks/]. The most common way to instantiate a framework is to inherit from some abstract classes defined in the framework hierarchy and write the code that is called by the framework itself. However, it is not always easy to identify which code is needed and where it should be written since frameworks class hierarchies can be very complex, especially for non-expert users.

Therefore, the "standard" instantiation process is not always the ideal way to describe a particular application. This happens because of several facts:

- the language used to build and use the framework is a wide spectrum language, in which it is not always easy to express the user intentions;
- the complexity of framework class hierarchies and the difficulty of finding the points where the code should be written: the framework hot-spots or flexible points [16, 19].

This work proposes a new approach for framework design and utilization that addresses these problems.

2. BACKGROUND AND RELATED WORK

Techniques such as design patterns [9], meta-object protocol [13], refactoring [12], and class reorganization [5] have been proposed to support framework design. However, they do not address the problem of helping the framework designer to find the commonalties (frozen spots) and flexible points (hot-spots) of a given framework. Our approach extends the idea of design patterns essentials [15, 16] to address this problem.

Domain specific languages (DSLs) are used to formally specify software designs [10]. A DSL is a formal language that is expressive over the abstractions of an application domain. An advantage of using a DSL is that the domain expert can express domain specific concepts in a formal way directly, instead of communicating the specification informally to a software specialist who is usually less familiar with the intended application.

An approach for using DSLs to enhance software libraries utilization is described in [17, 22]. The motivation for their work is that end users often do not make effective use of libraries, because most of them lack the expertise on properly identifying and composing the routines that form their application. They argue that in domains with mature subroutine libraries, one can greatly improve the productivity and quality of software engineering by automating the effective use of those libraries. Our work is an enhancement of this approach for frameworks.

3. APPROACH AND UNIQUENESS

We propose the use of formal methods, specifically the Z specification language [20, 21], to describe the semantics of the framework design structure. For this reason, we have developed an object-oriented meta-model in Z and will use this meta-model to describe all the artifacts and processes involved in our approach. There is a direct mapping between the syntax of our meta-model and the current object-oriented programming languages such as Java and C++.

Our approach to framework design is based on the idea that any design can be divided into two parts: the kernel sub-system design and the hot-spot sub-system design. The kernel sub-system design is common to all the applications that the framework may generate, and the hot-spot sub-system design describes the different characteristics of each application that can be supported by the framework. The hot-spot sub-system uses the method and the information provided by the kernel sub-system and may extend it. The hot-spot sub-system elements are the ones that vary for each application, and depend on the use of the framework. These elements must be adapted to each generated application.

Several unification rules, based on our meta-model, that show how frameworks can be derived from an initial set of applications have been defined as part of our work. These rules are used to structure and classify the framework architecture, highlighting the relationship between the kernel and hot-spot elements. Design pattern essentials [15, 16] are then applied to fulfill the flexibility requirements of each framework hot-spot.

We believe this approach will lead to a more explicit definition of the collaboration between the framework elements and will help the designer better implement the framework variability through the use of patterns.

Regarding framework utilization, we propose a different instantiation process, where a particular application is described by domain specific languages (DSLs) that are designed precisely for the framework domain. The technique basic idea is to capture domain concepts in DSLs, which will help the framework user to build code in an easier way, without worrying about implementation decisions and remaining focused on the problem domain. The specification written in each DSL is translated to the framework instantiation code through the use of transformational systems [14, 6].

A major topic in our work is the semi-automatic derivation of the DSLs from the framework design structure. The use of formal methods will allow us to prove two important theoretical results:

- there is always a mapping between frameworks and DSLs, which means that for every framework there are associated DSLs;
- the applications that can be derived from a framework through the "standard" instantiation process are exactly the same that can be specified as DSLs programs (using the associated, derived DSLs) .

4. RESULTS AND CONTRIBUTIONS

A set of desirable properties of a solution determines whether or not the solution is satisfactory. The proposed approach to enhance object-oriented frameworks design and utilization should posses the following properties.

- General: it should be general enough to be used in any application domain. This generality will be shown through case studies in different application areas;
- Precise: it should be precisely described, in order to be understood by its users and also to allow the definition of supporting environments. This property will be addressed through the formalization of the artifacts and processes involved in the approach;

Currently comparable methods are almost nonexistent or in their infancy. Case studies in three important and innovative application domains, in which we expect to contribute with the design and development of real frameworks, is also part of our work.

A number of papers [1, 2, 3, 4, 7, 8] report our current results. The consolidation of these research results will be present in my upcoming Ph.D. dissertation (July 99).

REFERENCES

1. P. Alencar, D. Cowan, T. Nelson, M. F. Fontoura, and C. J. Lucena, "Viewpoints and Frameworks in Component-Based Design", in *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, M. Fayad, R. Johnson, D. Schmidt (editors), John-Wiley, (to appear 1999).
2. P. Alencar, D. Cowan, S. Crespo, M. F. Fontoura, and C. J. Lucena, "Using Viewpoints to Derive a Conceptual Model for Web-Based Education Environments", MCC17/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to Journal of Systems and Software)*.
3. C. Braga, M. F. Fontoura, E. H. Haeusler, and C. J. Lucena, "Formalizing OO Frameworks and Framework Instantiation", First Brazilian Workshop on Formal Methods (WMF'98), UFRGS, 100-105, 1998.
4. C. Braga, M. F. Fontoura, C. J. Lucena, and L. M. Moura, "Using Domain Specific Languages to Instantiate OO Frameworks", MCC28/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to IEE Proceedings – Software Engineering).
5. E. Casais, "An incremental class reorganization approach", ECOOP'92, *volume 615 of Lecture Notes in Computer Science*, 114-132, 1992.
6. J. Cordy and I. Carmichael, "The TXL Programming Language Syntax and Informal Semantics", Technical Report, Queen's University at Kinkston, Canada, 1993.
7. M. F. Fontoura, L. M. Moura, S. Crespo, and C. J. Lucena, "ALADIN: An Architecture for Learningware Applications Design and Instantiation", MCC34/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to WWW Journal)*.
8. M. F. Fontoura, E. H. Hermann, and C. J. Lucena, "A Framework Design and Instantiation Method", MCC35/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998*.
9. E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, Design Patterns, *Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

10. P. Hudak, “Building Domain-Specific Embedded Languages”, *ACM Computing Surveys*, 28A(4), 1996.

11. R. Johnson, “Frameworks = (Components + Patterns)”, *Communications of the ACM*, 40(10), 1997.

12. R. Johnson and W. F. Opdyke, “Refactoring and aggregation”, Object Technologies for Advanced Software, First JSSST International Symposium, *volume 742 of Lecture Notes in Computer Science*, 264-278, 1993.

13. G. Kiczales, J. des Rivieres, and D. G. Bobrow, *The Art of Mataobject Protocol*, MIT Press, 1991.

14. J. C. S. P. Leite, M. Sant’anna, and F. G. Freitas, “Draco-PUC: a Technology Assembly for Domain Oriented Software Development”, 3rd IEEE International Conference of Software Reuse, 1994.

15. W. Pree, *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995.

16. W. Pree, *Framework Patterns*, Sigs Management Briefings, 1996.

17. Pressburger, T. and M. Lowry, “Automatic Domain-Oriented Software Design using Formal Methods”, Software Systems in Engineering, Energy-Sources Technology Conference and Exhibition, 33-42, 1995.

18. D. Roberts and R. Johnson, “Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks” in *Pattern Languages of Program Design 3*, Addison-Wesley, 1997.

19. H. A. Schmid, “Systematic Framework Design by Generalization”, *Communications of the ACM*, 40(10), 1997

20. J. M. Spivey, *Understanding Z: A Specification Language and Formal Semantics*, Cambridge University Press, 1988.

21. J. M. Spivey, *The Z Notation: A Reference Manual*, Prentice Hall, Hemel Hempstead, 1989.

22. Stickel, M., R. Walddinger, M. Lowry, T. Pressburger, and I. Underwook, “Deductive Composition of Astronomical Software from Subroutine Libraries”, 12th Conference on Automated Deduction, 1994.

23. R. Wirfs-Brock and R. Johnson, “Surveying current research in object-oriented design”, *Communications of the ACM*, 33(9), 1990.