USING VIEWPOINTS, FRAMEWORKS, AND DOMAIN-SPECIFIC LANGUAGES TO ENHANCE SOFTWARE REUSE

Sergio Crespo, Marcus Felipe M. C. da Fontoura, and Carlos José P. de Lucena
Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, 22453-900 Rio de Janeiro, Brazil
e-mail: [crespo, mafe, lucena]@inf.puc-rio.br

SUMMARY

Case studies have shown that high levels of software reuse can be achieved through the use of object-oriented frameworks. This paper describes a viewpoint-based design and instantiation method for framework development. This method uses the concept of viewpoints and the hot-spot relation in object-oriented design to guide the designer on the identification of hot-spots in the structure of the framework. Domain-specific languages are used to help the framework instantiation. A method supporting environment is also presented. Software design through the use of this environment appears to be a very successful approach from the point of view of component reuse.

KEY WORDS: object-oriented frameworks, viewpoints, domain-specific languages, framework design and instantiation, domain-oriented software development.

1. INTRODUCTION

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 in which the possibility 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 [10, 13]. 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. An object-oriented framework captures the common aspects of a family of applications, which can be seen as a domain theory [19].

Although object-oriented software development has experienced the benefits of using frameworks, a thorough understanding of how to identify, design, implement, and change them to meet evolving requirement needs is still object of research [24]. Techniques such as design patterns [10, 17], meta-object protocol [15], refactoring [14], and class reorganization [3] have been proposed to support framework development and cope with some of the challenges.

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.

2. FRAMEWORK DESIGN AND INSTANTIATION METHOD

In [19] Roberts and Johnson state that “Developing reusable frameworks cannot occur by simply sitting down and thinking about the problem domain. No one has the insight to come up with the proper abstractions.” They propose the development of concrete examples in order to understand the domain. The design strategy presented here is quite similar, analyzing concrete applications as viewpoints [8, 6] of a domain and deriving the final framework from the analysis of these viewpoints.

The 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 kernel structure is defined by analyzing the viewpoints design representations to produce a resulting design representation that reflects a structure that is common to all chosen viewpoints. This part of the design approach is based on a domain-dependent semantic analysis of the design diagrams to elicit the common features of the framework design structure.

The elements that are not in the kernel are the ones that vary for each application, and depend on the use of the framework. These elements define the framework hot-spots [17, 20] that must be adapted to each related application. A new relationship in object-oriented design, called the hot-spot relationship, has been defined to specify all the hot-spots in the system.

The semantics of this new relationship is given by the design patterns essentials [18]. This implies that the hot-spot relationship is in fact a meta-relationship that is implemented through a design pattern that is generated taking into account the hot-spot flexibility requirements. The hot-spot cards guides this generation process, providing a systematic way for generating design patterns based on flexibility properties.

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 and where it should be written since frameworks class hierarchies can be very complex, especially for non-expert users.

Therefore, the "common" 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, where 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, that are the framework hot-spots or flexible points.

We propose a different instantiation process [2], where a particular application is described through domain-specific languages (DSLs) [12] 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 the DSLs are translated to the framework instantiation code through the use of transformational systems [16, 4].

3. SUPPORTING ENVIRONMENT

A supporting environment that is used in the design and instantiation phases has been fully implemented (Figure 1). The environment helps all the method' steps:

1. Viewpoint definition: the users define the domain relevant viewpoints through the specification of OMT class diagrams;
2. Viewpoint unification: the environment applies the unification rules [9] to generate an abstract framework from the selected set of viewpoints;
3. Flexibility properties definition: flexibility requirements are defined to generate the appropriate design patterns that will compose the hot-spot sub-system structure;
4. DSLs definition: the definition of the syntax (in a BNF like notation) and semantics (through the specification of transformation rules) of the domain-specific languages used in the framework instantiation process. The Draco-PUC transformational system is used to perform the transformations and generate the framework instantiation code.

Through the use of this environment a framework for Web-based education (WBE), called ALADIN, was generated [5]. We used various existing WBE applications as viewpoints and derived the appropriate abstractions through the viewpoint unification process [1]. Two domain-specific languages were defined to instantiate the ALADIN framework: an educational language and a navigational language.

So far our studies show that the ALADIN framework can speed-up the development of WBE applications at least by a factor of 3. We are currently working in the electronic commerce domain to improve our investigation on the method (and on its supporting environment) completeness and effectiveness.

4. CONCLUSIONS

This paper shows how viewpoints, and particularly, the hot-spot relationship can be used as the main design driving force to reusable framework construction. The method provides mechanisms that help the framework developer in the identification of the kernel structure and the flexible parts.

The hot-spot card is a feature that can help us to bridge the gap between frameworks and design patterns, a still challenging issue [18]. The unification step is responsible for harmonically combining the various viewpoints. The use of the hot-spot relationship in this step helps the definition of the framework kernel and its hot-spots, introducing the necessary flexibility requirements.

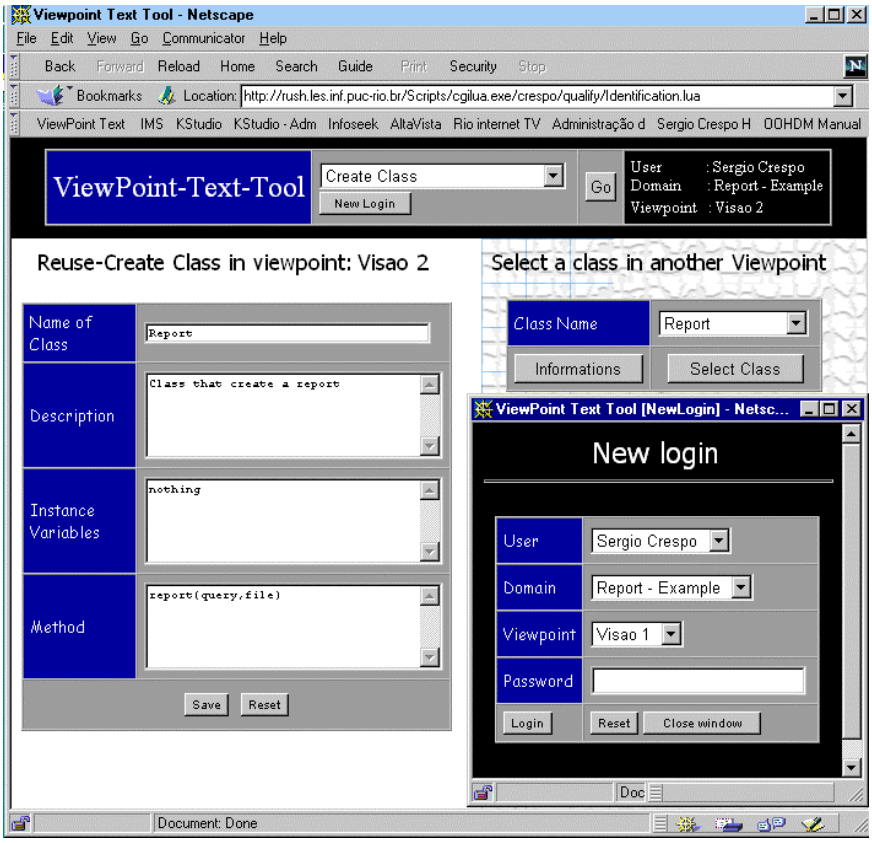

Figure 1. Method supporting environment

Another important characteristic of the method is that it is a complementary approach that can be combined to other techniques such as refactoring and OOADM. The method support environment presented in Section 3 uses the refactoring algorithms presented in [14] for the viewpoint unification process and the OMT notation for the viewpoint specification.

In [9] we present other examples of viewpoint unification and hot-spot instantiation. The method is currently completely formalized [9], based on the Z specification language [21, 22], object calculus [7], and category theory [11]. Through this formalization we could prove some important properties such as loose coupling.

In [1, 5] we present a case study for developing a framework for Web-based education. This framework is now fully implemented by the Software Engineering Laboratory, LES, of the Computer Science Department at PUC-Rio, Catholic University. An example of an application generated by this framework is the AulaNet™ environment (http://www.les.inf.puc-rio.br/aulanet)

REFERENCES

1. 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).

2. 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).

3. E. Casais, “An incremental class reorganization approach”, In ECOOP’92 Proceedings, volume 615 of Lecture Notes in Computer Science, pages 114-132, 1992.

4. J. Cordy and I. Carmichael, “The TXL Programming Language Syntax and Informal Semantics”, Technical Report, Queen’s University at Kinkston, Canada, 1993.

5. S. Crespo, M. F. Fontoura, C. J. Lucena, and L. M. Moura, “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).
6. J. Derrick, H. Bowman, and M. Steen, “Viewpoints and Objects”, Technical Report, Computing Laboratory, University of Kent, Canterbury, UK, 1997.
7. J. Fiadeiro and T. Maibaum, “Sometimes ‘Tomorrow’ is ‘Sometime’”, Temporal Logic, volume 827 of Lecture Notes in Artificial Intelligence, pages 48-66, Springer-Verlag, 1994.
8. A. Finkelstein and I. Sommerville, “The viwepoints faq” Software Engineering Journal, 11(1):2-4, January 1996.
9. 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.
10. E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, “Design Patterns, Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995.
11. R. Goldblatt, “Topoi – The Categorial Analysis of Logic”, North-Holland Publishing Company, 1979.
12. P. Hudak, “Building Domain-Specific Embedded Languages”, Computing Surveys, 28A(4), ACM, 1996.
13. R. Johnson, “Frameworks = (Components + Patterns)”, Communications of the ACM, Volume 40, Number 10, October 1997.
14. R. Johnson and W. F. Opdyke, “Refactoring and aggregation”, In Object Technologies for Advanced Software, First JSSST International Symposium, volume 742 of Lecture Notes in Computer Science, pages 264-278, Springer-Verlag, 1993.
15. G. Kiczales, J. des Rivieres, and D. G. Bobrow, “The Art of Mataobject Protocol”, MIT Press, 1991.
16. J. C. S. P. Leite, M. Sant’anna, and F. G. Freitas, “Draco-PUC: a Technology Assembly for Domain Oriented Software Development”; Proceedings of the 3rd IEEE International Conference of Software Reuse, 1994.
17. W. Pree, “Design Patterns for Object-Oriented Software Development”, Addison-Wesley, 1995.
18. W. Pree, “Framework Patterns”, Sigs Management Briefings, 1996.
19. 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.
20. H. A. Schmid, “Systematic Framework Design by Generalization”, Communications of the ACM, Volume 40, Number 10, October 1997
21. J. M. Spivey, “Understanding Z: A Specification Language and Formal Semantics”, Cambridge University Press, 1988.
22. J. M. Spivey, “The Z Notation: A Reference Manual”, Prentice Hall, Hemel Hempstead, 1989.
23. W. Turski and T. S. E Maibaum, “The Specification of Computer Programs”, Addison-Wesley Publishing Company, 1987.
24. R. Wirfs-Brock and R. Johnson, “Surveying current research in object-oriented design”, Communications of the ACM, 33(9), 1990.