A Prototype Software Engineering Environment for Domain Modeling and Reuse

H. Gomaa, L. Kerschberg, C. Bosch, V. Sugumaran, I. Tavakoli

Center for Software Systems Engineering
Department of Information and Software Systems Engineering
George Mason University
Fairfax, Virginia 22030-4444

Abstract

This paper describes a prototype software engineering environment which has been developed at George Mason University to demonstrate the concepts of domain modeling and software reuse. The prototype environment, which is application domain independent, is used to support the development of domain models and the generation of target system specifications from them. The prototype environment consists of an integrated set of commercial off-the-shelf software tools and custom developed software tools.

Keywords: domain modeling, reuse, software engineering environments, object repository, requirements elicitation, knowledge-based tool support

1 Introduction

At George Mason University, a project is underway to support software engineering lifecycles, methods, and prototyping environments to support software reuse at the requirements and design phases of the software lifecycle, in addition to the coding phase. A reuse-oriented software lifecycle, the Evolutionary Domain Lifecycle [Gomaa89, Gomaa91] has been proposed, which is a highly iterative lifecycle that takes an application domain perspective allowing the development of families of systems. A domain analysis and modeling method has also been developed [Gomaa90]. This paper describes the prototype software engineering environment that has been developed to demonstrate these concepts.

2 Domain Modeling

An application domain is defined to be a collection of systems that share common characteristics. A domain model is used to capture common characteristics and variations among a family of software

systems [Parnas79] in a given application domain. ¿From the domain model, target systems can be generated by tailoring the domain model. The concept of generating target systems from an application domain model has been adopted by various researchers [Batory89, Kang90, Lubars89, Pyster90].

Reuse is an important goal in domain modeling. The primary objective of the domain modeling approach to software development is to increase reuse, i.e., not only code reuse, but also of domain knowledge such as domain requirements, specifications and designs.

Applying the domain modeling method, the application domain is modeled by means of the following views:

- Aggregation Hierarchy. The Aggregation Hierarchy is used to decompose complex aggregate object types into less complex object types eventually leading to simple object types at the leaves of the hierarchy.
- Object Communication Diagrams. Objects in the real world are modeled as concurrent processes, which communicate with each other using messages. The object communication diagrams, which are hierarchically structured, show how objects communicate with each other.
- State Transition Diagrams. As each active object is modeled as a sequential process, it may be defined by means of a finite state machine and documented using a state transition diagram.
- Generalization / Specialization Hierarchies. As the requirements of a given object type are changed to meet the needs of a given target system, the object type may be specialized by adding, modifying or suppressing operations. The variants of a domain object type are stored in this hierarchy.
- Feature / Object Dependencies. This view shows for each feature (domain requirement) the object types required to support the feature.

The domain modeling method has been applied to developing a domain model for NASA's Payload Operations Control Center (POCC) Domain.

3 Prototype Software Engineering Environment

3.1 Overview

A prototype Software Engineering Environment is being developed, which consists of an integrated set of software tools that support domain modeling and the generation of target system specifications. In order to expedite development of the prototype, the environment uses commercial-of-the-shelf software as well as specially developed software. We are using IDE's Software Through Pictures CASE tool to represent the multiple views of the domain model, although semantically interpreting the views according to the domain modeling method. The information in the multiple views is extracted, checked for consistency, and stored in an object repository.

A knowledge based tool is used to assist with target system requirements elicitation and generation of the target system specification. The tool, implemented in NASA's CLIPS shell, conducts

a dialog with the human target system requirements engineer, prompting the engineer for target system specific information. The output of this tool is used to adapt the domain model to generate the target system specification.

The prototype Software Engineering Environment is a domain independent environment. Thus it may be used to support the development of a domain model for any application domain that has been analyzed, and to generate target system specifications from it.

3.2 Tool Support for Domain Modeling

The domain modeling activity starts with the creation of the multiple graphical views. The prototype environment uses a customized version of IDE's Software through Pictures (StP) as the front end to capture the multiple views. The StP environment was customized in two ways. First, the menu structure was changed to make it more consistent with the domain modeling method. The prototype menu consists of a set of icons, each corresponding to one of the views. Selecting each icon results in the invocation of the appropriate graphical editor for creating that view. Secondly, the schema in the underlying TROLL database was evolved by adding a new set of relations that capture the semantics of the domain model as represented in the graphical views. The relations serve as the interface between the front end graphical environment and the object repository, and they are independent of the StP environment.

We also extended the StP environment by integrating our own tools with it, in such a way that they could access the information in the underlying database as well as adding information to it. The Domain Model Relations Extractor (DMRE) extracts the information contained in the multiple views of the domain model from StP's TROLL relational database, interprets the information semantically according to the domain modeling method, and stores it in a common underlying relational representation. The Domain Model Relations Browser (DMRB) extracts, formats, and displays the domain model information in a relational format. The Domain Model Consistency Checker (DMCC) checks for consistency among the multiple views. Inconsistencies, if any, are displayed and must be corrected by the domain modeler. The entire process of creating the domain model relations and checking their consistency is repeated until a consistent set of relations is generated.

Once the relational representation has been determined to be consistent, the domain analyst uses the Domain Object Repository Generator tool to create a key component underlying the tools in our software engineering environment: the *object repository*. This repository is a single composite object that is composed of other objects representing domain object types, features, and the relationships among them which serve to define a domain model. The Domain Object Repository Generator tool takes the information captured in the relational representation and creates corresponding objects according to the object repository's schema. For example, if the domain analyst had created eight object communication diagrams using StP, the Domain Object Repository Generator tool would create eight instances of class OCD, the class defining object communication diagrams. Similarly, this tool will create objects representing the aggregation hierarchy, generalization/specialization hierarchies, and state transition diagrams, as well as the domain object types, external object types, and messages which are represented in these diagrams.

Another tool in our prototype software engineering environment supporting domain modeling is the Feature/Object Editor tool. Once the object repository representing a domain model has been

created, the domain analyst can use this tool to define new features by: 1) giving each new feature a unique name, 2) entering an informal annotation for each new feature, 3) specifying object types supporting the feature being defined, and 4) specifying other features required by the feature being defined. The Feature/Object Editor can also be used to browse features previously defined for a given domain model, delete features from the domain model, or modify the definition of features in a domain model.

The last tool supporting domain modeling in our prototype software engineering environment is the Domain Object Repository Report Generator tool. At any time after creating the object repository the domain analyst can generate a report on the contents of a given domain model by using this tool which extracts select information from the object repository as specified by the domain analyst, formats that information using the LaTeX typesetting program, and displays the resulting document in an X-Windows document previewer.

3.3 Tool Support for Generation of Target System Specifications

A target system specification is derived from the domain model by tailoring it according to the requirements specified for the target system. The process of generating a target system specification consists of gathering the requirements in terms of domain features, retrieving from the domain model the corresponding components to support those features, and reasoning about inter-feature and feature/object dependencies to ensure consistency. The Knowledge-Based Requirements Elicitation Tool (KBRET) has been developed to facilitate the process of generating target system specifications from the domain model. KBRET is implemented in CLIPS (C Language Integrated Production System), developed at NASA/Johnson Space Center.

The architecture of KBRET consists of two types of knowledge: domain independent and domain dependent knowledge. The domain independent knowledge provides control knowledge for the various functions supported by KBRET. These functions include a browser, a feature selector, a dependency checker, and a target system generator. The domain dependent knowledge represents the multiple views of an application domain model, including the feature/object dependencies. This knowledge is derived from the object repository by the Domain Dependent Knowledge Base Extractor tool, which structures this knowledge as CLIPS facts.

The separation of domain-independent and domain-dependent knowledge is essential for providing scale-up and maintainability of domain specifications for large domains. Also, since the domain-independent knowledge is independent of the application domain, it can be used with domain-dependent knowledge from any application domain to generate target system specifications in that domain.

To generate the target system specification, KBRET enters into a dialog with the target system engineer and elicits the requirements for the target system. KBRET also provides facilities for browsing the domain model. During the requirements elicitation process, KBRET presents the domain model features and the target system engineer can choose the features that are desired in the target system. Whenever a feature is selected for the target system, the associated feature/object dependencies are enforced. For example, when a feature with some prerequisite features is selected, those prerequisite features, if not already selected, are also included in the target system. Similarly, before deleting a feature from the target system, dependency checking is performed to ensure that it is not required by any other target system feature.

Once feature selection for the target system has been completed, KBRET begins the process of assembling the target system. The domain kernel object types are automatically included in the target system. Depending upon the features selected for the target system, the corresponding variant and optional object types are included. When all the object types required to support the selected features for the target system are assembled, the target system engineer is presented with the target system features and object types. KBRET also outputs two files containing the target system information that is used in tailoring the domain picture files to derive the multiple views for the target system.

Using the two files generated by KBRET, the Target System Specification Generator (TSSG) tool tailors the domain model graphical views and generates a set of graphical views for the target system. The target system views differ from those of the domain model in two ways. First, the optional objects that are not selected for the target system are removed. Secondly, in the case where one or more variants of a domain object type are selected, the object type is replaced by its variant(s).

4 Acknowledgements

We gratefully acknowledge the assistance of S. Bailin, R. Dutilly, J. M. Moore, and W. Truszkowski in providing us with information on the POCC. We gratefully acknowledge the major contributions of Liz O'Hara-Schettino in developing the domain model of the POCC. This work was sponsored primarily by NASA Goddard Space Flight Center, with support from the Virginia Center of Innovative Technology. The Software Through Pictures CASE tool was donated to GMU by Interactive Development Environments (IDE).

5 References

[Batory 89] Batory D., "The Genesis Database System Compiler: A Result of Domain Modeling", Proc. Workshop on Domain Modeling for Software Engineering, OOPSLA'89, New Orleans, October 1989.

[Gomaa 89] Gomaa H., R. Fairley and L. Kerschberg, "Towards an Evolutionary Domain Life Cycle Model", Proc. Workshop on Domain Modeling for Software Engineering, OOPSLA, New Orleans, October 3, 1989.

[Gomaa 90] Gomaa H., "A Domain Analysis and Specification Method for Software Reuse", Proc. Third Annual Workshop on Methods and Tools for Reuse, Syracuse, June 1990.

[Gomaa 91] Gomaa H. and L. Kerschberg, "An Evolutionary Domain Life Cycle Model for Domain Modeling and Target System Generation", Proc. Workshop on Domain Modeling for Software Engineering, International Conference on Software Engineering, Austin, May 1991.

[Kang 90] Kang K. C. et. al., "Feature-Oriented Domain Analysis", Technical Report No. CMU/SEI-90-TR-21, Software Engineering Institute, November 1990.

[Lubars 89] Lubars M. D., "Domain Analysis for Multiple Target Systems", Proc. Workshop on Domain Modeling for Software Engineering, OOPSLA'89, New Orleans, October 1989.

[Parnas 79] Parnas D., "Designing Software for Ease of Extension and Contraction", IEEE Transactions on Software Engineering, March 1979.

[Pyster 90] Pyster A., "The Synthesis Process for Software Development", in "System and Software Requirements Engineering", Edited by R. Thayer and M. Dorfman, IEEE Computer Society Press, 1990.

6 About the Authors

Hassan Gomaa is a Professor of Information and Software Systems Engineering at George Mason University, Fairfax, Virginia, where he teaches graduate courses in Software Systems Engineering. He also has several years industrial experience, most recently at General Electric.

He received his B.Sc. in Electrical Engineering from University College, London University, England and his Ph.D. in Computer Science from Imperial College, London University. His current research interests include software engineering methods and tools, analysis and design of concurrent and real-time systems, software prototyping, software process models, domain analysis and design, and software reuse. He has published over fifty technical papers.

Larry Kerschberg is Professor and Chairman of the Department of Information and Software Systems Engineering in the School of Information Technology and Engineering at George Mason University. He holds a Ph.D. in Systems Engineering from Case Western Reserve University, an M.Sc. in Electrical Engineering from the University of Wisconsin-Madison, and a B.Sc. degree in Engineering Science from Case Institute of Technology.

Dr. Kerschberg's research focuses on the areas of data models, database design, data dictionaries, distributed query processing, and most recently, expert database systems. His current research interests deal with architectures for expert database systems, models that integrate the specification of knowledge and data, as well as knowledge-based tools and techniques for knowledge acquisition and software requirements gathering and specification.

Dr. Kerschberg serves as an Editor-in-Chief of the forthcoming International Journal of Intelligent Information Systems.

Chris Bosch is a doctoral candidate studying Information Technology at George Mason University where his dissertation research focuses on problems associated with the evolution of object-oriented systems. He holds a B.S. in Aerospace Engineering from the University of Virginia (1982) and an M.A. in Government from Georgetown University (1985), and is a member of ACM and AAAI.

Vijayan Sugumaran is a doctoral candidate in Information Technology at George Mason University. He received his B.Tech. and M.S. degrees in Mining Engineering from Indian School of Mines and University of Alaska-Fairbanks respectively. His research interest includes database management systems, knowledge-based systems, domain modeling and software engineering.

Iraj Tavakoli received the BS in computer science from Aryamehr Univeristy in Tehran/Iran in 1980, and the MS in computer science from University of Tennessee in 1988. He is currently a PH.D student in School of Information Technology at George Mason University. His research interests include software reuse, software engineering environments, domain modeling, and real-time system design.