Methods and Tools for Domain Specific Software Architectures

Hassan Gomaa

Department of Information and Software Systems Engineering
George Mason University
Fairfax, Virginia, 22030-4444
Tel: (703) 993-1652
Email: hgomaa@isse.gmu.edu

Abstract

At George Mason University, a project is underway to support software engineering lifecycles, methods, and 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 [1, 2], has been proposed, which is a highly iterative lifecycle that takes an application domain perspective allowing the development of families of systems. Current emphasis is on the domain analysis and specification phase of the EDLC for developing an application domain model, which captures the similarities and variations of the domain.

Keywords: domain specific software architectures, evolutionary domain lifecycle, system families.

Workshop Goals: To understand how others are approaching the domain specific software architecture development problem.

Working Groups: reuse process models, domain analysis/engineering.

1 Position

1.1 The Evolutionary Domain Life Cycle

The Evolutionary Domain Life Cycle (EDLC) Model is a software life cycle model that eliminates the traditional distinction between software development and maintenance. Instead, systems evolve through several iterations. Furthermore, because new software systems are often outgrowths of existing ones, the EDLC model takes an application domain perspective allowing the development of families of systems. The EDLC model incorporates two related sub-life cycles, domain modeling and target system generation. Domain modeling deals with developing the reusable requirements and domain specific software architecture for a family of systems, while target system generation deals with generating target systems from the domain model.

A Domain Model is a problem-oriented architecture for the application domain that reflects the similarities and variations of the members of the domain. Given a domain model of an application domain, an individual target system (one of the members of the family) is created by tailoring the domain model given the requirements of the individual system.

1.2 Domain Modeling

In a domain model, an application domain is represented by means of multiple views, such that each view presents a different aspect of the domain [3]. The different views are developed iteratively.

1. 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. Objects types are either kernel, i.e., required in every target system, or optional, i.e., only required in some target systems. The Aggregation Hierarchy supports the IS-PART- OF relationship.

2. Object communication diagrams. Objects in the real world are modelled as concurrent tasks, which communicate with each other using messages. Dynamic relationships between objects, in the form of messages passed between objects, are shown by means of object communication diagrams.

3. State transition diagrams. Since each object is modeled as a sequential task, an object may be defined by means of a state transition diagram, whose execution is by definition strictly sequential.

4. Generalization / Specialization Hierarchy. As the requirements of a given object type are changed to meet the specific needs of a target system, the object type may be specialized by adding, modifying or suppressing operations. In domain modeling, the variants of a domain object type are stored in a Generalization / Specialization Hierarchy (GSH), which supports the IS-A relationship.

5. Feature / Object dependencies. This view relates the end-user's perspective of the domain, namely the features supported by the domain, to the object types in the domain model. It shows for each feature (domain requirement) the object types required to support the feature. Also defined are any prerequisite features required and any mutually exclusive features. This view is particularly important for optional features, since it is the selection of the optional

features, and the object types required to support them, that determine the nature of the desired target system.

1.3 Generation of Target System Specification

A Target System Specification is a problem-oriented architecture for an individual target system, i.e., a member of the family of systems that constitute the domain. It is a tailored instance of the Domain Model. Requirements are reused by selecting those features required in the target system and then tailoring the domain model, subject to the appropriate feature object dependency constraints, to generate the target system specification.

1.4 Domain Modelling Environment

A proof-of-concept experiment has also been carried out to develop a prototype domain modeling environment [Gomaa91b], which consists of an integrated set of software tools that support domain modeling and target system requirements elicitation. The environment uses commercial-of-the-shelf software as well as custom developed software. The graphical editors supported by the Software Through Pictures CASE tool are used to represent the multiple views of the domain model, namely the Aggregation Hierarchy, the Object Communication Diagrams, the Generalization / Specialization Hierarchies and the State Transition Diagrams. However, the multiple views are semantically interpreted according to the domain modeling method. The information in the multiple views is extracted, checked for consistency, and mapped to an object repository. The feature / object dependencies are defined using a Feature / Object Editor.

A knowledge based tool is used to assist with target system requirements elicitation and generation of the target system specification [4]. 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 multiple views of the target system specification.

The prototype 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.

1.5 Current Status

A major issue in the domain modeling approach to developing domain specific software architectures concerns how effectively it addresses scaleup. The problem is not so much the size of the domain (which is a "common" software engineering problem) but the degree of variability and volatility in the domain. A relatively stable well understood application domain is likely to be the best candidate for domain modeling. The method has been applied to a real-world problem, NASA's Payload Operations Control Center (POCC) domain.

There is likely to be some tradeoff between scaleup and the amount of variation allowed in a domain model. Too much variation is liable to lead to a combinatorial explosion. We are currently exploring the concept of feature packages as an approach to scaleup, where features are grouped and treated as one super feature. The current approach is oriented towards concurrent systems, including real-time and distributed systems. We are also adding an information modeling view, to address more information intensive applications. Also under investigation are the design and implementation phases of the EDLC [5].

1.6 Acknowledgements

The author is indebted to his research colleagues and students L. Kerschberg, R. Fairley, C. Bosch, E. O'Hara-Schettino, V. Sugumaran, and I. Tavakoli, for their invaluable assistance throughout. This work was sponsored in part by NASA Goddard Space Flight Center and the Virginia Center of Innovative Technology. The Software Through Pictures CASE tool was donated to GMU by Interactive Development Environments.

References

[1] H. Gomaa, R. Fairley, and L. Kerschberg, "Towards an evolutionary domain life cycle model," in *OOPSLA'89, New Orleans*, Oct. 1989.

[2] G. H. and L. Kerschberg, "An evolutionary domain life cycle model for domain modeling and target system generation," in *Proceedings of the Workshop on Domain Modeling for Software Engineering, International Conference on Software Engineering, Austin, TX*, May 1991.

[3] H. Gomaa, "An object-oriented domain analysis and modeling method for software reuse," in *Proceedings of the Hawaii International Conference on System Sciences*, Jan. 1992.

[4] L. Gomaa H., Kerschberg and V. Sugumaran, "A knowledge-based approach for generating target system specifications from a domain model," in *Proceedings of the IFIP World Computer Congress, Madrid, Spain*, Sept. 1992.

[5] H. Gomaa, "A reuse-oriented approach for structuring and configuring distributed applications," *Software Engineering Journal*, Mar. 1993.