AGL: Incorporating Behavioral Aspects into Domain-Driven Design

Duc-Hanh Dang[a,b,*], Duc Minh Le[c], Van-Vinh Le[a]

[a] *Department of Software Engineering, VNU University of Engineering and Technology, Vietnam*
[b] *Vietnam National University, Hanoi*
[c] *Department of Information Technology, Swinburne Vietnam, FPT Univeristy*

Abstract

Context: Domain-driven design (DDD) aims to iteratively develop software around a realistic domain model. Recent works in DDD have been focusing on using annotation-based domain-specific languages (aDSLs) to build the domain model. However, within these works behavioral aspects, that are often represented using UML Activity and State machine diagrams, are not explicitly captured in the domain model.
Objective: This paper focuses on defining a novel unified domain modeling method in order to integrate behavioral aspects into domain models following the DDD approach. Specifically, behavioral aspects as part of a unified domain model are represented using a new aDSL, named activity graph language (AGL). Such an incorporation of the AGL and the previously-developed aDSL (DCSL) for a unified domain model would allow us to achieve three important features of a DDD: feasibility, productivity, and understandability.
Method: Our method consists in constructing a configured unified domain model within a domain-driven architecture. We used the annotation attachment feature of the host programming language like Java to attach AGL's activity graph directly to the activity class of the unified model, thereby, creating a configured unified model. The abstract and concrete syntax of AGL are also defined in this work. We demonstrate our method with a Java framework named jDomainApp and evaluate AGL using a case study to show that it is essentially expressive and usable for real-world software.
Results: This work brings out (1) a mechanism to incorporate behavior aspects for a unified domain model, in which a new aDSL named AGL is defined to represent the domain behaviors; and (2) a unified modeling method for domain-driven software development.
Conclusion: Our method significantly extends the state-of-the-art in DDD in two important fronts: constructing a unified domain model for both structural and behavioral aspects of domain models and bridging the gaps between model and code.

Keywords: Domain-driven design (DDD); Module-based Architecture; Domain-specific language (DSL); UML/OCL-based domain modelling; Attribute-oriented Programming (AtOP)

*Corresponding author
Email addresses: hanhdd@vnu.edu.vn (Duc-Hanh Dang), duclm20@fe.edu.vn (Duc Minh Le), vinhskv@gmail.com (Van-Vinh Le)