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 diagrams and statecharts, 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, we aim to develop a new aDSL, named activity graph language (AGL), that allows us to capture behavioral aspects, and then, incorporate the language with our previously-developed aDSL, named DCSL, for a unified domain model with three important features: 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 OOPL to attach an AGL's activity graph directly to the activity class of the unified model, thereby, creating a configured unified model. We adopt the UML/OCL meta-modeling approach to specify the abstract and concrete syntax of AGL. We demonstrate our method with an implementation in 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) the AGL as an aDSL to express the domain behaviors; (2) a mechanism to incorporate behavior aspects for a unified domain model; and (3) 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)