An Evaluation of Ontology Based Domain Analysis for Model Driven Development

Hong Guo, Anhui University, Hefei, China
Shang Gao, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
John Krogstie, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
Hallvard Trætteberg, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
Alf Inge Wang, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

ABSTRACT

Although Domain Analysis (DA) is important for Model Driven Development (MDD), traditional DA methods are demanding and not practical in many situations. When computer games are developed, game design (problem domain) is usually decided in a gradual way within iterations where software prototypes are constructed and playtest are performed. In such a case, it is not practical to fit a heavyweight DA in the highly iterative process. Researchers indicated that vocabularies were expected to automate game design. Such vocabularies can be reused in another form in DA tasks. In this research, the authors developed an ontology and a DA procedure based on it. To evaluate them, theoretical analysis, case studies, and a user acceptance survey were used. The results indicated that the ontology met the general requirement as a domain vocabulary, and it enhanced the DA process in an expected way. Most of external potential users (46 in total) considered the ontology useful and easy to use.

Keywords: Computer Game, Domain Analysis, Domain Specific Modeling (DSM), Game Development, Model Driven (Software) Development (MDD), Ontology, Pervasive Game

1. INTRODUCTION

Model Driven (Software) Development (MDD) is useful to handle domain complexity, shorten software development cycle and improve software quality. The successful application of MDD relies heavily on the Domain Analysis (DA) task as it produces essential domain artifacts for MDD use. Formal DA showed good design result but the usage of formal DA methods was still limited
(Ceh, Crepinšek, Kosar, & Mernik, 2011). This is because such methods are very demanding and often not practical. For instance, when computer games are developed, game design (problem domain) cannot be defined before the software development. Instead, it is defined together with the software development. Numerous iterations are employed where (software) prototypes are constructed and playtest are performed to validate the design. In such a process, a more lightweight and flexible DA method is expected. Researchers investigated Ontology Based Domain Analysis (OBDA) (Tairas, Mernik, & Gray, 2009). By using OBDA, existing ontologies are used in another form to ease the DA process. In this research, we followed a similar direction. We developed an ontology and a DA procedure based on the ontology. To make our work more concrete, instead of focusing on general computer games, we focus on pervasive (computer) games. The rest of this paper is organized as follows: Section 2 introduces background of MDD, DA methods, as well as their application in (pervasive) the computer game domain. Section 3 illustrates the Pervasive Game Ontology (PerGO) as well as the DA procedure. Section 4 evaluates PerGO and the DA method. And at last, we conclude the paper and point out some future research directions in Section 5.

2. BACKGROUND

In this section, we introduce some background knowledge.

2.1. MDD and DA

Using models has been a long-standing tradition to design complex software systems, and it has become even more popular after the Unified Modeling Language (UML) was developed (Fowler, 2004). Models help us understand a complex problem and its possible solutions. However, they have been perceived primarily as documentation artifacts, and thus the creation and usage of them has seemed to be peripheral to software development (France & Rumpe, 2007). This situation was changed by the application of MDD, where models (rather than computer programs) became the primary focus and products of software development (Selic, 2003).

The motivation of MDD is to move the working focus from programming to solution modeling (Stahl, Vlter, & Czarnecki, 2006). This is achieved by two important mechanisms: providing abstractions that are close to the problem domain and generating programs from their corresponding models (Stahl et al., 2006). To play to its strength, MDD should be domain specific (Kelly & Tolvanen, 2008). This requires Domain Specific Languages (DSLs) to raise the level of abstraction, domain specific code generators to automate codes generation, as well as domain specific platforms to reduce the complexity of the code generators.

DSL development usually takes four stages (decision, analysis, design, and implementation) (Mernik, Heering, & Sloane, 2005). Domain analysis helps meet two major challenges of DSL: the abstraction challenge (how to provide support for creating and manipulating problem-level abstractions) and the formality challenge (how to formalize the semantics and what aspects of semantics need to be formalized) (France & Rumpe, 2007). Three outputs are usually generated from DA: domain definition, domain specific vocabulary with semantic meanings, and a model describing commonality and variability space. These outputs contribute to not only the construction of DSLs, but also the construction of code generators (Mernik et al., 2005) and platforms. Domain analysis supports decision making up front, and provides concrete inputs for design and implementation afterwards. As a result, the quality of DA is critical for the overall effectiveness and efficiency of MDD application.
A Novel Query-Driven Clustering-Based Technique for Vertical Fragmentation and Allocation in Distributed Database Systems
<www.igi-global.com/article/a-novel-query-driven-clustering-based-technique-for-vertical-fragmentation-and-allocation-in-distributed-database-systems/176732?camid=4v1a>

RapidOWL: A Methodology for Enabling Social Semantic Collaboration
<www.igi-global.com/chapter/rapidowl-methodology-enabling-social-semantic/28856?camid=4v1a>