Chapter 8
Productivity Analysis of the Distributed QoS Modeling Language

Joe Hoffert
Vanderbilt University, USA

Douglas C. Schmidt
Vanderbilt University, USA

Aniruddha Gokhale
Vanderbilt University, USA

ABSTRACT

Model-driven engineering (MDE), in general, and Domain-Specific Modeling Languages (DSMLs), in particular, are increasingly used to manage the complexity of developing applications in various domains. Although many DSML benefits are qualitative (e.g., ease of use, familiarity of domain concepts), there is a need to quantitatively demonstrate the benefits of DSMLs (e.g., quantify when DSMLs provide savings in development time) to simplify comparison and evaluation. This chapter describes how the authors conducted productivity analysis for the Distributed Quality-of-Service (QoS) Modeling Language (DQML). Their analysis shows (1) the significant productivity gain using DQML compared with alternative methods when configuring application entities and (2) the viability of quantitative productivity metrics for DSMLs.

INTRODUCTION

Model-driven engineering (MDE) helps address the problems of designing, implementing, and integrating applications (Schmidt, 2006; Hailpern, 2006; Atkinson 2003; Kent, 2002). MDE is increasingly used in domains involving modeling software components, developing embedded software systems, and configuring quality-of-service (QoS) policies. Key benefits of MDE include (1) raising the level of abstraction to alleviate accidental complexities of low-level and heterogeneous software platforms, (2) more effectively expressing designer intent for concepts in a domain, and (3) enforcing domain-specific development constraints. Many documented benefits of MDE

DOI: 10.4018/978-1-61692-874-2.ch008

are qualitative, *e.g.*, use of domain-specific entities and associations that are familiar to domain experts, and visual programming interfaces where developers can manipulate icons representing domain-specific entities to simplify development. There is a lack of documented quantitative benefits for domain-specific modeling languages (DSMLs), however, that show how developers are more productive using MDE tools and how development using DSMLs yields fewer bugs.

Conventional techniques for quantifying the benefits of MDE in general (*e.g.*, comparing user-perceived usefulness of measurements for development complexity (Abrahao and Poels, 2007, 2009)) and DSMLs in particular (*e.g.*, comparing elapsed development time for a domain expert with and without the use of the DSML (Loyall, Ye, Shapiro, Neema, Mahadevan, Abdelwahed, Koets, & Varner, 2004)) involve labor-intensive and time-consuming experiments. For example, control and experimental groups of developers may be tasked to complete a development activity during which metrics are collected (*e.g.*, number of defects, time required to complete various tasks). These metrics also often require the analysis of domain experts, who may be unavailable in many production systems.

Even though DSML developers are typically responsible for showing productivity gains, they often lack the resources to demonstrate the quantitative benefits of their tools. One way to address this issue is via *productivity analysis*, which is a lightweight approach to quantitatively evaluating DSMLs that measures how productive developers are, and quantitatively exploring factors that influence productivity (Boehm, 1987; Premraj, Shepperd, Kitchenham, & Forselius, 2005). This chapter applies quantitative productivity measurement using a case study of the *Distributed QoS Modeling Language* (DQML), which is a DSML for designing valid QoS policy configurations and transforming the configurations into correct-by-construction implementations. Our productivity analysis of DQML shows significant productivity gains compared with common alternatives, such as manual development using third-generation programming languages. While this chapter focuses on DQML, in general the productivity gains and analysis presented are representative of DSMLs' ability to reduce accidental complexity and increase reusability.

BACKGROUND

This section presents related work in the area of metrics for MDE and domain-specific technologies. We present work on quantitative analysis for MDE technologies as well as metrics to support quantitative evaluation.

Conway and Edwards (2004) focus on measuring quantifiable code size improvements using the *NDL Device Language* (NDL), which is a domain-specific language applicable to device drivers. NDL abstracts details of the device resources and constructs used to describe common device driver operations. The creators of NDL show quantitatively that NDL reduces code size of a semantically correct device driver by more than 50% with only a slight impact on performance. While quantifiable code size improvements are shown by using NDL, the type of improvement is applicable to DSLs where a higher level language is developed to bundle or encapsulate lower level, tedious, and error prone development. The productivity analysis for a DSL is easier to quantify since common units such as lines of source code are used. Conway and Edwards present compelling evidence of productivity gains of NDL although they do not encompass all the benefits of automatic code generation found with DSMLs such as the ease of a GUI.

Bettin (2002) measures productivity for domain-specific modeling techniques within the domain of object-oriented user interfaces. Comparisons are made between (1) traditional software development where no explicit modeling is performed, (2) standard Unified Modeling