Impact of MDE Approaches on the Maintainability of Web Applications: An Experimental Evaluation*

Yulkeidi Martínez¹, Cristina Cachero², Maristella Matera³, Silvia Abrahao⁴, and Sergio Luján²

¹ Universidad Máximo Gómez Báez de Ciego de Ávila, Cuba
² Universidad de Alicante, Spain
³ Politecnico di Milano, Italy
⁴ Universidad Politécnica de Valencia, Spain

Abstract. Model-driven Engineering (MDE) approaches are often recognized as a solution to palliate the complexity of software maintainability tasks. However, there is no empirical evidence of their benefits and limitations with respect to code-based maintainability practices. To fill this gap, this paper illustrates the results of an empirical study, involving 44 subjects, in which we compared an MDE methodology, WebML, and a code-based methodology, based on PHP, with respect to the performance and satisfaction of junior software developers while executing analyzability, corrective and perfective maintainability tasks on Web applications. Results show that the involved subjects performed better with WebML than with PHP, although they showed a slight preference towards tackling maintainability tasks directly on the source code. Our study also aims at providing a replicable laboratory package that can be used to assess the maintainability of different development methods.

1 Introduction

It is well known that maintenance is the most expensive phase in the software life cycle, absorbing between 45% and 60% [1] of total management costs. This situation has led many researchers to focus on maintainability from different perspectives. On the one hand, unstructured maintenance postulates wading straight into the code of the target application in order to make the necessary changes, normally with the aid of specialized tools. Sometimes this process relies on automatic-maintainability measures that operate over the code of the target application [2]. On the other hand, structured maintenance examines and modifies the design, and then -either manually or automatically- reworks the code to match it. Many authors claim that structured maintenance is a more reliable and efficient process than unstructured maintenance [3].

* The authors wish to thank the students who kindly agreed to participate in this empirical study.
The Model Driven Engineering (MDE) paradigm goes one step further in the lane of structured maintenance, and advocates the use of models and model transformations to speed up and simplify the maintenance process. The MDE community claims several advantages over code-based maintenance processes, such as short and long-term productivity gains, improved project communication and defect and rework reduction [4,5]. Unfortunately, despite all these claims and the fact that MDE practices are maturing by the day, practitioners still lack a body of practical evidence that soundly backs the purported maintainability gains due to the use of this paradigm [5,6]. As Glass says, “We (practitioners) need your help. We need some better advice on how and when to use methodologies” [7]. Such evidence can be provided in the shape of empirical studies—such as surveys, experiments, case studies or postmortem analysis studies—that directly measure the effect of the chosen paradigm on the developer performance and satisfaction [8]. Many authors have written about the importance of providing empirical evidence in Software Engineering (SE) [9,10]. Unfortunately, the percentage of empirical studies on maintainability in MDE approaches is still very low, contrasting with other disciplines and even other areas of SE [11]. This paper tries to fill this gap, and presents the results of a quasi-experiment in which two groups of junior developers performed a set of maintainability tasks on two Web applications.

The paper is structured as follows: Section 2 presents the rationale behind the selection of WebML and PHP as typical examples of the two current paradigms (model-driven and code-based) in Web applications development, as well as the definition of maintainability that we are using all along the paper, and how it has been assessed in the past in MDE methodologies. This background sets the context for the definition of hypotheses and variables in Section 3 together with the experimental design (subjects, instrumentation, operation and data collection mechanisms). Section 4 presents the data analysis and an interpretation of results that takes into account the identified threats to validity. Last, Section 5 concludes the paper and presents some further lines of research.

2 Background

The last years have witnessed a continuous evolution in the scenario of Web application development. A number of technologies have been proposed, fostering the adoption of server-side and client-side languages for the Web. Within this plethora of proposals, some technologies have been largely adopted by practitioners. Server-side scripting is one such technology, through which the generation of HTML markup is achieved by means of languages like JSP, PHP, and ASP.NET. Among them, numbers point at PHP as the most popular server-side scripting language today [12]. Given such diffusion, in our experiment we have chosen it as representative of the programming languages for the Web.

In parallel to the technology innovation line, the Web Engineering community has devoted several efforts to the definition of MDE approaches [13], strongly characterized by the adoption of conceptual models. Such approaches offer high-level abstractions, capturing the most salient characteristics of Web applications,