Information and Software Technology 62 (2015) 164–186

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

In search of evidence for model-driven development claims: An experiment on quality, effort, productivity and satisfaction

Jose Ignacio Panach [a,*], Sergio España [b], Óscar Dieste [c], Óscar Pastor [b], Natalia Juristo [c]

[a] Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València, Avenida de la Universidad, s/n, 46100 Burjassot, Valencia, Spain
[b] Centro de Investigación en Métodos de Producción de Software – ProS, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain
[c] Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Spain

ARTICLE INFO

Article history:
Received 3 October 2014
Received in revised form 19 February 2015
Accepted 21 February 2015
Available online 28 February 2015

Keywords:
Automatic programming
Methodologies
Programming paradigms
Quality analysis and evaluation

ABSTRACT

Context: Model-Driven Development (MDD) is a paradigm that prescribes building conceptual models that abstractly represent the system and generating code from these models through transformation rules. The literature is rife with claims about the benefits of MDD, but they are hardly supported by evidences.
Objective: This experimental investigation aims to verify some of the most cited benefits of MDD.
Method: We run an experiment on a small set of classes using student subjects to compare the quality, effort, productivity and satisfaction of traditional development and MDD. The experiment participants built two web applications from scratch, one where the developers implement the code by hand and another using an industrial MDD tool that automatically generates the code from a conceptual model.
Results: Outcomes show that there are no significant differences between both methods with regard to effort, productivity and satisfaction, although quality in MDD is more robust to small variations in problem complexity. We discuss possible explanations for these results.
Conclusions: For small systems and less programming-experienced subjects, MDD does not always yield better results than a traditional method, even regarding effort and productivity. This contradicts some previous statements about MDD advantages. The benefits of developing a system with MDD appear to depend on certain characteristics of the development context.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Model-Driven Development (MDD) [15,33] is a paradigm advocating the use of models as the primary software development artefact and model transformations as the main operation. The idea is that all that is needed to develop a system is to build its conceptual model [37]. The conceptual model is the input to a model compiler that automatically generates software code to implement the system, or to a model interpreter that directly executes the model. MDD is the natural continuation of the evolution that gradually raised the abstraction level from assembly languages to third-generation programming languages [40].

Although MDD recommends automating as much code generation as possible, nowadays there is a wide range of approaches to apply the paradigm. Some of the proposals, such as OO-Method [40] WebRatio [6], Genexus [1] and OOHDM [41], generate fully functional systems through automatic transformations. Others generate part of the system. For example, NDT [24] can generate all of the code that supports behaviour and persistency, but most of the user interface needs to be manually implemented.

MDD advocates often claim that it has advantages over traditional software development. For example, Mellor [33] states that the use of models increases productivity, and Selic [42] states that MDD helps to improve productivity and reliability. However, few of these claims have been empirically evaluated. Existent empirical studies focus on measuring time, overlooking other characteristics that MDD is claimed to have, such as quality. There is a lack of empirical evaluations of MDD, probably due to the inherent complexity of comparative evaluations of software development methods and the challenges of adopting MDD in industrial contexts. Staron has reported the following difficulties suffered when applying MDD under conditions of practice [46]: (i) MDD methodological and technological learning curves are high, (ii) there is no development standard, (iii) relations among the multiple views within the conceptual model are unclear, and (iv) the transformations needed to generate code from models are difficult to design.

* Corresponding author.
E-mail addresses: joigpana@uv.es (J.I. Panach), sergio.espana@pros.upv.es (S. España), odieste@fi.upm.es (Ó. Dieste), opastor@pros.upv.es (Ó. Pastor), natalia@fi.upm.es (N. Juristo).

http://dx.doi.org/10.1016/j.infsof.2015.02.012
0950-5849/© 2015 Elsevier B.V. All rights reserved.

In this work, we have designed and conducted an experiment to verify some of the claimed advantages of MDD. We aim to contribute to corroborating or refuting some of the claims that have been historically attributed to MDD and widely published in the literature. We have compared an MDD with a traditional method where developers implement the code manually. The experimental tasks are to develop small but fully-functional web applications from scratch. We focus on evaluating software quality and developer effort, productivity and satisfaction since these are the most popular claims about MDD in the literature. The experimental subjects are last-year master students who have competence in traditional development and no significant previous experience with MDD. As operationalisation of MDD, we have used an industrial tool that can generate fully-functional systems from conceptual models: INTEGRANOVA [2]. MDD is applicable to the development of any system, such as information [40], embedded [47] or cyber-physical systems [27], among others. Our experiment focuses on information systems.

For inexperienced developers, we have observed that there are no significant differences between MDD and a traditional method regarding effort, productivity and satisfaction. However, we have observed that quality in MDD is more stable than a traditional method to variations in problem complexity. These preliminary results clearly contradict the claims that have been accepted as facts (i.e. quality, effort, productivity and satisfaction in MDD are always better) and call for a thorough study and deeper understanding of the conditions under which MDD might be better than other development paradigms. We have analysed some reasons why MDD claims are not satisfied in our experiment. We have identified some variables that appear to influence the suitability of the development paradigm (MDD or traditional) to a project situation, such as problem complexity and developers' background experience with MDD. Results must be interpreted within the context in which the experiment has been run: (i) the subjects are students, (ii) they have previous experience with traditional development and they are learning to develop information systems with MDD, (iii) the systems are developed from scratch and (iv) their size is small. We conclude that further experimental research is required to gain insight and to better understand the conditions under which MDD might be an alternative to traditional software development.

The paper is organised as follows. Section 2 discusses related work. Section 3 describes the experiment definition and planning. Section 4 presents the outcomes of the study. Section 5 shows the threats to validity identified after running the experiment. Section 6 discusses the interpretation of the results. Finally, Section 7 shows the conclusions.

2. Related work

We have reviewed the literature in search of statements claiming benefits of MDD. We have generalised similar statements from different works and grouped those statements that refer to the same topic, as seen below:

S1. Improvements in coding and in the resulting code:
- S1.1 Improvement of software code quality [44,10,34].
- S1.2 Reduction of flaws in software architecture [4].
- S1.3 Improvement of code consistency [4,10].
- S1.4 Rapid code generation when the application needs to be deployed on distinct platforms [31,4] or migrated from one platform to another as technology changes [44].
- S1.5 Automatic application of tested software blueprints and industry-standard patterns [10].
- S1.6 Elimination of repetitive coding for the application [44].

S2. Improvements related to models:
- S2.1 Models are always updated with the code [19].
- S2.2 The model becomes the focus of development effort; it is no longer discarded at the outset of coding [44].

S3. Improvements in maintenance:
- S3.1 Improvement in reuse, development of new versions and maintainability [19].
- S3.2 Reduction of intellectual effort required for understanding the system [42,34].
- S3.3 Mappings provide interoperability among two or more different platforms [44,17].

S4. Improvements for developers:
- S4.1 Reduction in developer effort [44,4,10,19,43,42].
- S4.2 Improvement of productivity [42,11,34].
- S4.3 Enhancement of developer satisfaction [30].

Some of these advantages are embedded in the very definition of MDD and have no need of experimental validation. For example:

- Code generation for distinct platforms: the same model can be used to derive code for different programming languages or platforms [44,31,4].
- Improvement of code quality: developers do not need special skills to build a good architecture [4].
- Maintainability: if a programming language evolves, the model compiler can be updated with a new version of the code and the developer can effortlessly update the code from the same model automatically [19].

However, most of the claims listed above can be subject to experimental investigation. There exist some works that have gathered empirical evidences about MDD. Some authors have defined specific frameworks to guide the *evaluation of non-trivial MDD advantages*. For instance, Vanderose and Habra [48] define a framework that explicitly includes the various models used during software development as well as their relationships in terms of quality. Since generic frameworks [50,9] have been widely validated and are frequently used in software engineering to evaluate different types of technology, the benefits of using specific frameworks to evaluate MDD are unclear.

MDD has been adopted by some companies. Some authors have described their *experience of applying MDD in industrial settings*. Baker et al. [8] report on 15 years of applying MDD at Motorola and analyse effort, quality and productivity in automatic code generation and automatic test generation. According to their results, effort is 2.3 times less, defects are between 1.2 and 4 times less, and productivity is between 2 and 8 times greater using co-simulation, automatic code generation and model testing.

Some authors aim to extract the *existent experience with MDD at companies*. For example, Hutchinson et al. [21] focus on understanding which factors lead to a successful adoption of MDD. They interviewed 20 professionals by telephone. The participants were from three different companies: a printer company, a car company, and a telecom company. They found that: MDD requires a progressive and iterative approach; successful MDD adoption depends on organisational commitment; MDD users must be motivated to use the new approach; an organisation using MDD needs to adapt its own processes along the way; MDD must have a business focus, where MDD is adopted as a solution to new commercial and organisational challenges.

Notice that both self-experience and surveys elicit opinions, so their findings are empirical but, by definition, subjective.

Other researchers have conducted *case studies* to identify MDD benefits. Mellegard and Staron [32] performed a case study to compare whether developers expend more effort on modelling in MDD

Download English Version:

https://daneshyari.com/en/article/551663

Download Persian Version:

https://daneshyari.com/article/551663

Daneshyari.com