Computers in Industry 96 (2018) 86–97

Contents lists available at ScienceDirect

Computers in Industry

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

BRCode: An interpretive model-driven engineering approach for enterprise applications

Anderson Oliveira, Vinicius Bischoff, Lucian José Gonçales*, Kleinner Farias, Matheus Segalotto

University of Vale do Rio dos Sinos (UNISINOS), 950, Unisinos Av. – Postal Address: 93.022-000, São Leopoldo, Rio Grande do Sul, Brazil

ARTICLE INFO

Article history:
Received 2 June 2017
Received in revised form 22 November 2017
Accepted 9 January 2018
Available online xxx

Keywords:
Interpretive MDE
Industry
Case study
Productivity
Profitability

ABSTRACT

Many model-driven engineering (MDE) approaches have been proposed in recent studies. They claim to improve software quality and productivity by raising the abstraction level at which developers work. However, they often fall short of what was expected in terms of productivity, profitability, and Return on Investment in real-world scenarios. This article proposes BRCode, which is an interpretive MDE approach for fast-changing enterprise applications. A case study that involves the development of an Enterprise Resource Planning (ERP) system enabled data collection based on 34 realistic scenarios in a Brazilian company. This evaluation compared BRCode with a generative MDE (genMDE) approach. Our results show that (1) genMDE required 93.75% more effort; and (2) genMDE and BRCode led to financial gains in 48% and 70% of the cases, respectively. On average, genMDE led to financial losses in most cases, while BRCode roughly tripled financial gains; (3) BRCode had an ROI of 1.54, compared to 0.07 for genMDE, which represents a difference of 93.37%. The results were encouraging and show the potential for using BRCode to support software production companies in the turbulent business environment.

1. Introduction

The development of enterprise applications has occurred in increasingly unstable business environments in industry. Usually, complex and fast-changing customer requirements, pressures of shorter development cycles, and rapidly advancing information technologies are ever-present characteristics of most mainstream projects [1]. In this turbulent context, software production companies seek to use generative or interpretive model-driven engineering (MDE) approaches for improving software quality and developer productivity [2,3]. In [4], Hailpern and Tarr highlight that these benefits might be achieved by raising the abstraction level at which developers work.

Generative and interpretive MDE approaches are functionally equivalent. Generative MDE approaches (genMDE) aim at transforming model-to-model or model-to-code, while interpretive MDE approaches focus on interpreting models or meta-data to produce run-time applications [2]. Both approaches claim to improve software quality and productivity by raising the abstraction level at which developers work. However, they often fall short of what would be expected in terms of productivity, profitability, and Return on Investment in real-world scenarios [5]. Still, little has been reported about their effectiveness in mainstream projects in industry. Even worse, current MDE approaches have not provided stable architectures for supporting software products in increasingly turbulent business environments. This instability goes beyond the limits of inconvenience to developers and customers. The rework that is caused by constant changes leads to a cost increase, which is often not cheap.

Today, countless works report that the greater the number of changes, the greater the likelihood that flaws and defects may arise in software products [6]. Usually, the entire software architecture must be changed to accommodate changes that would not affect core requirements of an enterprise application. Despite this, MDE approaches have been adopted, without empirical evidence about their benefits or side-effects. For example, these approaches may fail to increase the capability of the development team to develop, ensure quality, and release enterprise systems more quickly, while ensuring low development cost.

* Corresponding author.
E-mail addresses: andersonmo@edu.unisinos.br (A. Oliveira), viniciusbischof@edu.unisinos.br (V. Bischoff), lucianj@edu.unisinos.br (L.J. Gonçales), kleinnerfarias@unisinos.br (K. Farias), msegalotto@edu.unisinos.br (M. Segalotto).

https://doi.org/10.1016/j.compind.2018.01.002
0166-3615/

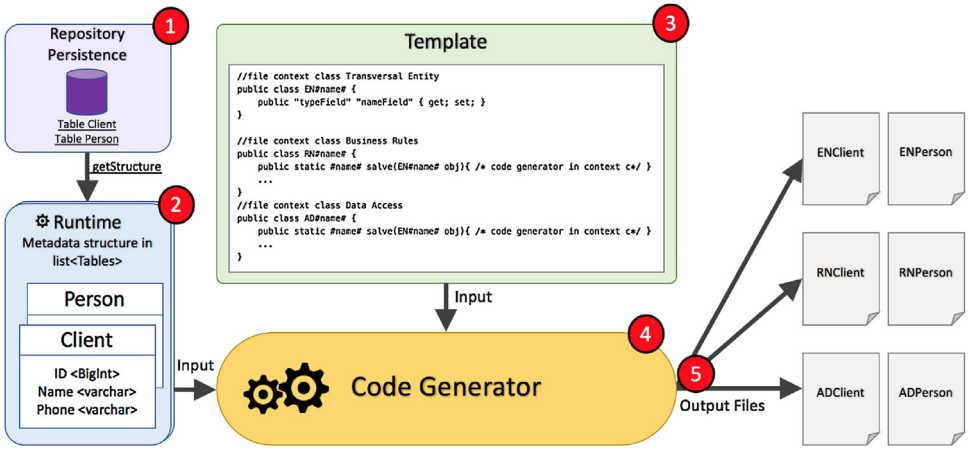

Fig. 1. The generative MDE approach that was adopted by the company.

This article, therefore, proposes BRCode, which is an interpretive MDE approach for fast-changing enterprise applications. Software developers benefit from using the BRCode approach typically when performing development and maintenance tasks, such as rendering data-driven user interfaces, elaborating architectural design, and delivering fast, low-cost enterprise applications. For this, BRCode provides a set of key features that leverage the productivity of software developers by reusing ever-present features in enterprise applications. For example, by automatically rendering user interfaces from meta-data, developers might invest more time designing databases and studying the business rules more carefully.

A case study that involves development of an Enterprise Resource Planning (ERP) system enabled data collection based on 34 realistic scenarios in a Brazilian company. Our results show that (1) genMDE required 93.75% more effort; (2) genMDE and BRCode led to financial gains in 48% and 70% of the cases, respectively. On average, genMDE led to financial losses in most cases, while BRCode roughly tripled financial gains; (3) BRCode had an ROI of 1.54, compared to 0.07 for genMDE, which represents a difference of 93.37%. The obtained results are encouraging and show the potential for using BRCode to support software production companies in volatile business environments. Moreover, this study is the first to perform an empirical study that compares generative versus interpretive MDE approaches in real-world settings. The empirical knowledge and insights that are generated may serve as a basis for improving the current MDE approaches.

The remainder of this article is organized as follows. Section 2 describes the generative MDE approach that is used in our case study. Section 3 introduces the BRCode and describes its architecture and main features. Section 4 describes how the proposed approach was evaluated through an empirical study in industry. Section 5 presents the collected results. Section 6 contrasts this study with the current literature. Section 7 presents some conclusions and discusses future studies.

2. Generative MDE approach

MDE approaches [4] aim at shifting the development focus from code to models. Existing MDE approaches are based on *code generation* or *model interpretation* [2]. Approaches of the first type use models and transformers to represent high-level concepts of abstraction and carry out sets of transformations, respectively. The transformers can convert abstract models into less abstract models (i.e., model-to-model transformations) or directly into source code that might be compiled (i.e., model-to-text transformations) [7]. The source code that is generated from models might cover most of the platform's complexity and configuration details. Consequently, software developers end up concentrating their time on problems that are related to business rules, for example. MDE approaches of the second type use run-time interpretation to produce software that conforms to the model. Typically, these MDE approaches produce an interpreter, which is embedded in the application.

We will mainly focus our attention on the description of the generative MDE approach that was adopted by the Brazilian company in which BRCode was evaluated (Section 4). To facilitate understanding of how this technique was used, Fig. 1 presents an overall scheme that shows its main elements, which are described as follows:

1. **Repository:** Software developers produce a data repository based on the customer's requirements and create domain models from these requirements. These models are pivotal to creating the database of the application that is under development. The *Repository* can be seen Fig. 1(1).
2. **Runtime:** The application is created based on meta-data that are present in the database. Operations, such as CRUD[1] (Create, Read, Update, and Delete), and business rules are defined. Transformers generate tables at the database and data-access objects to manipulate these tables. The *Runtime* component is shown in Fig. 1(2).
3. **Template:** Software developers can personalize user interface components, such as data masks, combo boxes, styles, object positions, effects, data fields, and database calls. They can also configure how the application manages the information inputs and transforms the retrieved data into a readable format for the clients of the application. The *Template* component is exhibited in Fig. 1(3).
4. **Code Generator:** The application structure is generated based on the template and run-time inputs. For this, the code generator converts these inputs into source code. This component, which is shown in Fig. 1(4), plays a central role in supporting the generative MDE approach that is used in our case study (discussed in Section 4).
5. **Output Files:** These files are responsible by representing the generated files throughout the generative process. The application structure is generated, including the entity, business rules,

[1] The four basic functions of persistent storage: create, read, update, and delete (as an acronym CRUD).