The Journal of Systems and Software 128 (2017) 106–129

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A tool to support the definition and enactment of model-driven migration processes

Fco. Javier Bermúdez Ruiz*, Óscar Sánchez Ramón, Jesús García Molina

Faculty of Informatics, University of Murcia, Campus of Espinardo, 30100, Murcia (Spain)

A R T I C L E I N F O

Article history:
Received 4 April 2016
Revised 8 March 2017
Accepted 9 March 2017
Available online 10 March 2017

Keywords:
Model-driven engineering
Software processes
Software migrations
Process enactment

A B S T R A C T

One of the main challenges to achieve the industrial adoption of Model-Driven Engineering (MDE) paradigm is building tools able to support model-driven software processes. We present a tool for the definition and enactment of model-driven migration processes. We have created a SPEM-based language for defining Abstract Migration models that represent an MDE migration solution for a particular pair of source and target technologies. For each legacy application to be migrated, the Abstract Migration model is transformed into a Concrete Migration model which contains all the information needed for the enactment. Then, these models are enacted by means of a process interpreter which generates Trac tickets for executing automated tasks by means of Ant scripts and managing manual tasks with the Mylyn tool.

Our work has therefore two main contributions: i) it proposes a novel solution for the enactment that integrates the execution of the automated tasks with the generation of tickets to support the manual tasks, and ii) it describes how MDE techniques can be used to implement process engineering tools, in particular migration processes. The article presents the approach and describes in detail the essential aspects of our tool.

1. Introduction

Software modernisation typically refers to understanding and evolving existing software assets to maintain their business value. A legacy system is modernised when maintenance is not enough to achieve the desired improvements (e.g., new capabilities or greater maintainability) since that system must be extensively changed. Software migration is a form of modernisation that involves moving an application, as a whole or a part of it, from the platform on which is currently operating to a target platform that provides better features. A migration can be done in a disciplined way by applying a software re-engineering process that consists of three stages: reverse engineering, restructuring, and forward engineering (Seacord et al., 2003).

Model-Driven Software Engineering (MDSE or simply MDE) has emerged as a new area of software engineering that emphasises the systematic use of models in the software lifecycle in order to improve its productivity and software quality aspects such as maintainability and interoperability. MDE techniques, e.g. metamodelling and model transformations, allow tackling the complexity of software by raising its abstraction and automation levels (Brambilla et al., 2012). These techniques have been proven useful not only for developing new software applications (OMG, 2003; Kelly and Tolvanen, 2008) but also for modernising legacy systems. In the latest years, MDE techniques have been applied to a variety of modernisation scenarios (OMG, 2008a; Ulrich and Newcomb, 2010), especially in the migration of applications (Fleurey et al., 2007; Ramón et al., 2014). However, building tools for supporting MDE software processes is a challenge that must be met to achieve the industrial adoption of MDE (Selic, 2012).

As Leon J. Osterweil stated in his influential paper (Osterweil, 1987) about the nature of software processes, *"software processes are software too"*, so they can be described by specifications (i.e. models) that can be executable. *Process Engineering* (Gruhn, 2002) is the Software Engineering area focused on the modelling and enactment of process models. MDE techniques can significantly leverage this area as some works recently presented have illustrated. Most of the activity has been focused on the SPEM metamodel (OMG, 2006) and the definition of approaches to enact SPEM models (Bendraou et al., 2005; 2007; Ellner et al., 2010). How MDE development processes can be supported by using MDE-based process engineering tools has received little attention up to date (Golra and Dagnat, 2012; Koudri and Champeau, 2010; Steudel et al., 2012; Gamboa and Syriani, 2016; Maciel et al., 2013).

* Corresponding author.
E-mail addresses: fjavier@um.es (Fco.J. Bermúdez Ruiz), osanchez@um.es (Ó. Sánchez Ramón), jmolina@um.es (J. García Molina).

http://dx.doi.org/10.1016/j.jss.2017.03.009
0164-1212/

MDE software processes integrate automated tasks (e.g. model-to-text transformations) with tasks to be manually performed by developers (e.g. writing code for the business logic layer). A process engineering tool supporting such processes should provide basic functionality such as: i) the specification of the software process, ii) the execution of the automated tasks, iii) the support and guidance for the software managers and developers involved in the manual tasks, and iv) the integration of manual and automated tasks into a task workflow.

Our research group collaborated with a software development company in a pilot project aimed at migrating Oracle Forms applications to the Java platform. In this project, model-driven re-engineering was applied to partly automate the migration effort. Due to the lack of software environments with the aforementioned functionality, we built the Models4Migration tool described in this article. Unlike when MDE techniques are used to develop new software, MDE migration processes involves repeatedly applying a model transformation chain to all the existing artefacts of the same kind (e.g., DDL scripts and GUI code). As stated in Feiler and Humphrey (1992) “It is desirable to define software processes with sufficient precision so that many of the routine enactment tasks can be automated”. In the case of a model-driven migration, models to be enacted should provide information on concrete artefacts of the legacy application to be migrated. In addition, the MDE migration experience described in Fleurey et al. (2007) evidenced that “to maximise the efficiency of the migration process, the tasks that are left to the developer have to be clearly identified and the developer should be provided with all the information he or she needs”. These specificities of MDE migrations have been considered in building our tool, which is based on an MDE approach that has been implemented around three main design choices: i) The definition of a SPEM-based language tailored to express MDE migration processes; ii) A migration is defined at two levels of modelling: abstract models represent a migration process for a pair of source and target technologies and they are refined into concrete models that include the information needed to be enacted; iii) The enactment of a concrete model consists of automatically executing automated tasks and generating manual tasks as Trac[1] tickets which are managed as Mylyn[2] tasks. Team leaders and developers could save a great effort with the proposed automation.

Some approaches have been proposed to enact MDE processes (Steudel et al., 2012; Gamboa and Syriani, 2016; Maciel et al., 2013). However, they do not support the aforementioned specific requirements of MDE migrations. Therefore our work presents two main contributions. Firstly, the manual task interface with Trac server is one of the main novelties of the approach. Ticket creation is very useful in order to implement manual tasks owing to it is able to define the context for guiding the task completion inside well-known development environment, such as Eclipse. However, creating tickets is a tedious and time consuming task to be performed by team leaders. We have defined an enactment approach that automatically generates these tickets and this generation is integrated with the execution of automated tasks. This automation is specially useful for MDE migration processes. We have chosen Trac[3] and Mylyn[4] since they are open-source tools commonly used by software companies, but tools with similar functionality could be used in our approach. Note that our approach goes beyond the definition of software processes provided by tools such as EPF,[5] or the enactment proposed in some approaches which does not support the execution of applications which addresses the tasks of a process (Ellner et al., 2012; Golra and Dagnat, 2012; Koudri and Champeau, 2010).

Secondly, our work shows how an MDE approach can be used to build a tool supporting software development processes, in particular MDE-based migration processes, from the definition of software processes to the management of the tasks to be performed by managers and developers. Note that this article is focused on the migration tool built to support the definition and enactment of migration processes, being the details of the actual migration processes left out.

The paper is organised as follows. The next section introduces some basic concepts about migration processes, model-driven engineering, and the SPEM metamodel; in addition, some issues that arise when dealing with model-driven migration processes are addressed. Section 3 presents the running example that will be used to illustrate the proposed approach, which is outlined in Section 4. The following three sections explain in detail each one of the tasks that are supported by the Models4Migration tool, i.e. the definition, instantiation and enactment of migration models. Section 8 will describe how the migration tool can be used and Section 9 will show how has been applied to a real case study. Next, some lessons learned are commented in Section 10. Finally, the related work is presented in Section 11 and the conclusions are drawn in Section 12.

2. Background and motivation

The aim of this section is to motivate our work and introduce some background about software process modelling and model-driven migrations. First, we analyse some essential aspects of software process modelling and elicit the functionality to be provided by a tool supporting a model-driven migration. Next, we define some basic concepts of MDE and introduce the SPEM language. Finally we present a black box vision of the tool created, showing the inputs and the outputs of the tool.

2.1. Modelling and enactment of migration processes

A software process involves the accomplishment of a *workflow of activities* which create the software artefacts of the target system. Each of these activities can be composed of several tasks which indicate how to fulfil them. For example, the migration of procedures that implement business logic can be done by performing an automatic translation by some means, or by a development team that implement them by hand. Tasks can be classified accordingly to the way they are accomplished in three categories (Osterweil, 1987):

- *Automated tasks*: the goal of the task can be achieved without any human intervention, usually by the execution of one or more tools. For instance, the execution of model transformations by means of a model transformation engine.
- *Manual tasks*: the goal of the task must be achieved by a human, either because it is difficult to automate or because it requires supervision. For instance, a code completion task where a developer has to implement some functionality.
- *Semi-automated tasks*: the goal of the task is achieved in a partly automated way as it requires human performance or interaction at some point. For instance, an assistant window which requires some data from an developer to complete some functionality.

It is interesting to differentiate that activities show *what* to do, and the tasks show *how* to do it. Therefore, note that activities are more abstract concepts than tasks. It is also worth noting that activities as well as tasks must be arranged in order for the process to be analysed or executed.

[1] http://trac.edgewall.org.
[2] http://www.eclipse.org/mylyn.
[3] https://trac.edgewall.org/.
[4] http://www.eclipse.org/mylyn/.
[5] http://www.eclipse.org/epf.