Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 321 (2016) 41–65

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Evolution of a Model-driven Process Framework

Wilson Pádua[1,2]

Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil

Abstract

We discuss the evolution of Praxis, a model-driven process framework, building on feedback from educational and professional applications, along the past fifteen years. We follow the evolution from Praxis first version to the current one, discussing what was introduced in each. For past and current versions, we classify model improvements, discussing their nature and rationale, derived from received feedback.

Keywords: process, model-driven development, model transformations, CRUD transactions, framework, reuse, persistent data.

1 Introduction

According to the CMMI [10], a defined software development process has a maintained process description, and contributes process related experiences to the organizational process assets. By process framework we define a set of artifacts which includes process descriptions and other important kinds of assets, such as reusable libraries and guidance resources.

Such framework is defined as model-driven when models are its core artifacts, from which others are partially or completely derived. In this work, we describe how a model-driven framework evolved along fifteen years, through improvements suggested by feedback from both educational and professional applications.

The process framework whose evolution is discussed here is Praxis, whose primary purpose is to support software engineering course projects. As such, it has been used in the last fifteen years to support teaching in software engineering courses.

[1] We thank IBM Rational for supporting this work, within the IBM Academic Initiative.

[2] Email: wppf@ieee.org

http://dx.doi.org/10.1016/j.entcs.2016.02.004

Moreover, Praxis has been systematically applied and evaluated by the author himself, in industry-oriented, graduate software courses. The results of this kind of application have been discussed elsewhere ([30], [31], [32], [33]). As shown there in more detail, the students in such courses were required to develop small applications using the complete process. Typical courses comprised four software engineering disciplines, with about 30 hours each, and typical student project had a size of about 100 to 150 function points.

Praxis-Synergia, a derived process tailored to real-life projects, has been applied in the development of applications in the range of hundreds to thousands of function points. This application is performed by Synergia, a university-based software engineering laboratory which develops real-life applications under contract, mostly for government organizations ([35], [5]).

The Praxis process has evolved along those years, mostly through feedback from both course and Synergia projects. This paper describes which changes were introduced during those years, as feedback from process use was collected and analyzed.

In section 2, we discuss the goals of process and modeling improvements, proposing a classification for them. In section 3, we present the evolution of a model-driven development process, oriented to support course projects, showing the improvements performed in each version, how such improvements were suggested by feedback from its application, and which kinds of change they caused. In section 4, we discuss current work. Conclusions are drawn in section 5.

2 Process Improvements

2.1 Improvement goals

A major process improvement goal is to make it more **effective**, that is, help projects to accomplish their mission within specified constraints. For real-life projects, this usually means delivering a product with a satisfactory quality level, meeting the product requirements in a provable way. However, even a fully effective process will not allow competitive development if it is not also **efficient**, accomplishing such mission within its market budget and schedule constraints. In the evolution of processes, feedback from process application leads to actions to improve both the process effectiveness and efficiency

For educational processes, effectiveness also means exercising the knowledge and skills that their application in course projects intends to impart. Efficiency also means keeping those projects within course budgets for time and effort.

2.2 Process artifacts

A software development process aims to produce **executable code**, such as application code and test scripts, together with their **environmental data**, such as database schemata, test data, configuration files, localized text, graphics and other resource files.

A number of other artifacts help delivering such code and data. Some may be