beContent: a model-driven platform for designing and maintaining Web applications[*]

Antonio Cicchetti[1], Davide Di Ruscio[2], Romina Eramo[2], Francesco Maccarrone[2], and Alfonso Pierantonio[2]

[1] School of Innovation, Design and Engineering
Mälardalen University,
SE-721 23, Västerås, Sweden
antonio.cicchetti@mdh.se

[2] Dipartimento di Informatica
Università degli Studi dell'Aquila
Via Vetoio, Coppito I-67010, L'Aquila, Italy
{diruscio|romina.eramo|francesco.maccarrone|alfonso}@di.univaq.it

Abstract. Model Driven Engineering (MDE) is increasingly gaining acceptance in the development of Web Applications as a mean to leverage abstraction and render business logic resilient to technological changes. This paper describes the beContent project with its modeling languages and tools, which aims at the automated generation of rich Web applications.

1 Introduction

The beContent project [1] aims at defining an infrastructure (see Figure 1) consisting of a coordinated collection of languages and tools which permit to shorten systems' life-cycle and ease maintainence tasks. The gluing element of the project is the beContent metamodel (BMM) which is based on a previous work of the authors [2]. The metamodel defines the abstract syntax of the modeling languages: a diagrammatic and a textual concrete syntax, called beContent modeling language (BML) and beContent textual language (BTL), respectively, endowed with a round-tripping mechanism. In other words, they can interchangeably be used for specifying a system and, for instance, whenever a diagrammatic specification undergoes a modification the textual counterpart is consistently updated and the other way round. This has been possible by using the AMMA framework [3] and an additional component such as GMF [4]. The models can be edited by means of a visual and textual editor realized as Eclipse plugins as described in Sect. 3.

2 beContent Models

A beContent model consists mainly of the declarative and coordinated specification of three different concerns:
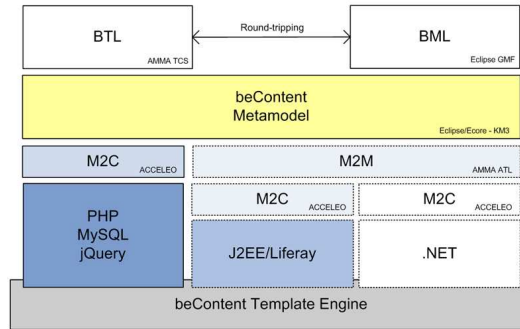

Fig. 1. The beContent infrastructure

- the *data* view is the description of the relational model of the data, in essence it describes the metadata of the application;
- the *content* view describes the data sources and how the content is retrieved and aggregated in pages; and finally
- the *interaction* view specifies how to manipulate the information entered in the forms (e.g., a textual content can be edited by means of text fields, textareas, or rich-text editors), the validation constraints, and additional information which may affect the interaction between the user and the application.

Fig. 2. Sample *news* model

An example of specification is given in Figure 2, where a simple description of *news* is illustrated as a composition of three main model elements: the entity, the form, and the content. In particular, the form is contained in a *manager page* which indicates that the code for managing all the interactions (creation, update, and deletion) are encapsulated in an individual page. Starting from models conforming to BMM *model-to-code* transformations written in Acceleo [5] are capable of generating the whole corresponding applications. The generated artifacts and the model transformations are illustrated in the next section.

3 The beContent framework

The framework consists of a number of tools which have been implemented (or are planned to be implemented) on Eclipse EMF [6] and are listed in table 1 together with the specific EMF technology being used and their maturity status.

Description	Technology	Status
BMM	AMMA KM3	*advanced*
BML	GMF	*advanced*
BTL	AMMA TCS	*incubation*
J2EE/Liferay code generation	Acceleo	*incubation*
LAMP code generation	Acceleo	*advanced*
Model Repository	EMF Teneo	*preliminary*
Versioning + Co-evolution	EMF Compare, AMMA ATL	*incubation*
Workflow Metamodel	AMMA KM3	*preliminary*

Table 1. Platform components and related technologies

In particular, BMM has been defined by means of KM3, a metamodeling language part of the AMMA framework. As aforementioned, models are edited by means of a visual and textual editors: the former is an Eclipse plugin (realized with GMF) which assists the designer in editing the diagrammatic descriptions; whereas the latter is defined using TCS which uses text-to-model trace-links that are created during parsing to allow hyperlinks and hovers for references within the text. Once created, the models can be mapped onto several platforms (see Figure 1) by means of automated transformations which generate the corresponding assets. The model-to-code transformations have been written by means of the Acceleo templating language, which provides enough flexibility and model navigation functionalities. The platforms currently supported are the typical LAMP and J2EE/Liferay portal platform, although the latter is still in a preliminary stage of the development. As an example, consider the model in Figure 2, which can be transformed by the LAMP model-to-code to obtain the functionalities depicted in Figure 3. All the tools have been released under an open source license scheme and can be freely downloaded at [1].
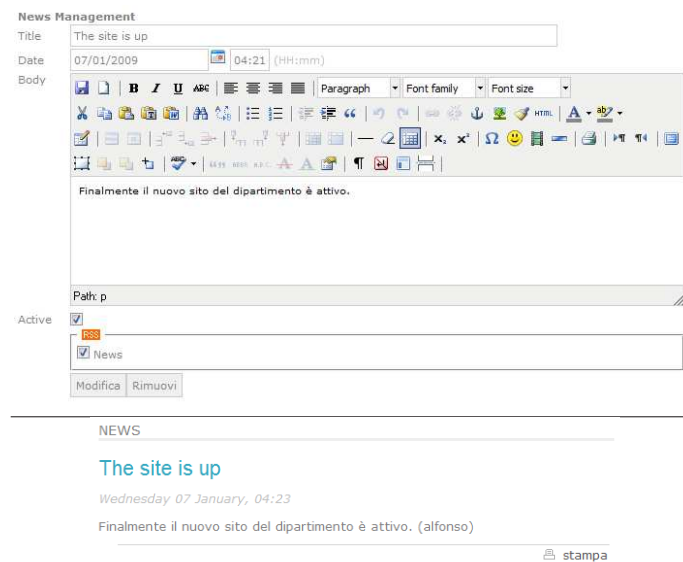

Fig. 3. Generated *news* interaction and content

4   Additional features

Additional aspects of the platform include an advanced *user management* which is reflected in the metamodel and mapped on the choosen target platform. Moreover, users are allowed to interact one with another by means of processes which are described conforming to the *workflow* metamodel extension which is currently under development. The topology of processes are given by means of a restriction of UML activity diagrams (in essence, it is limited to non parallel processes) which are mapped to specific workflow engines integrated with the corresponding platforms.

Analogously to any other software artifacts, Web applications are subject to a heavy evolutionary pressure. Therefore, we are currently designing and implementing a specific support to *model repository*, *versioning*, and *co-evolution*. In particular, as models are considered *first-class* artefacts, they can also be compared to detect the changes a model underwent during its life-cycle. This provides beContent with an evolution support able to *a)* calculate by means of EMF Compare the differences between two versions of the same model, *b)* represent the evolution as a model, and *c)* automatically generate from this model a refactoring procedure which migrates the application and its data (schema and contents) [7].

5   Conclusions and future work

This paper described a number of aspects of the beContent project. The approach has been validated on the LAMP platform by producing the following sites

- `http://www.abruzzo24ore.tv`, a news portal whose features include multimedia and rich-text content, registered users, RSS feeds, and forums;
- `http://www.di.univaq.it`, the official site of the Computer Science Department at the University of L'Aquila where the project has been initiated;
- `http://www.univaq.it`, the official site of the University of L'Aquila.

We also briefly illustrated a transformational approach to the co-evolution of applications, i.e., when certain aspects of a model changes then other components are requested to be adapted to remain valid. Future works includes the implementation refinement/enhancement and how to map the beContent metamodel to well-known frameworks, such as *rails*, *django*, and *spring*.

References

1. Pierantonio, A.: beContent. http://www.becontent.org (2006)
2. Di Ruscio, D., Muccini, H., Pierantonio, A.: A Data Modeling Approach to Web Application Synthesis. Int. Jour. of Web Engineering and Technology **1**(3) (2004) 320–337
3. Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P.: Modeling in the Large and Modeling in the Small. In: European MDAFA Workshops. Volume 3599 of LNCS. (2004) 33–46
4. Eclipse project: GMF - Graphical Modeling Framework. http:// www.eclipse.org/gmf/
5. Obeo: Acceleo. http://www.acceleo.org
6. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Addison Wesley (2003)
7. Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.: Automating co-evolution in model-driven engineering. In: 12th IEEE Intl. EDOC Conference. (2008) 222–231