Ontology, rule and DSL-based software re-engineering: a case study

Nuno Silva[1], Paulo Maio[1], Alexandre Gouveia[1], Evaristo Figueiredo[1], and Rui Lopes[2]

[1] School of Engineering, Polytechnic of Porto, 4249-015 Porto, Portugal
{nps,pam,aas,evmfe}@isep.ipp.pt
[2] DigitalWind, Lda., 3810-498 Aveiro, Portugal
rlopes@digitalwind.pt

Abstract. uebe.Q is a managing software for solid referential information systems, such as ISO 9000 (for quality) and ISO 1400 (for environment). This is a long-term developed software, encompassing extensive and solid business logic with a long and successful record of deployments. A recent business model change imposed that the evolution and configuration of the software, shifts from the company (and especially the development team) to consultants and other business partners. The so far acceptable rigidity, fragility, immobility and opacity of the software became a problem. This paper describes the approach adopted in re-engineering the software by using (i) ontologies for the specification of business concepts, (ii) closed-world assumption (CWA) rules for the specification of the dynamics of the system and (iii) DSL for the configuration of the system by domain/business experts.

Keywords: Software re-engineering · Ontology · Rules · Semantic WS.

1 Introduction

uebe.Q[3] is an information managing software for solid referential information systems, such as ISO 9000 (for quality) and ISO 1400 (for environment). This is a long-term developed software, encompassing extensive and solid business logic with a long and successful record of deployments in different business areas and countries. Currently and for the last decade, when uebe.Q software is deployed for/on a customer, an extensive and specific configuration process takes places involving the customer's business experts and the DigitalWind's software development and operation teams. So, typically, DigitalWind is the developer, the seller and the business consultant altogether.

A recent business model change determined that the configuration and adaptation tasks of the software shifts from the DigitalWind's development and operation teams to third-party consulting and other business partners.

However, uebe.Q configuration and deployment requires strong business and technological expertise, thus constraining the adoption of the new business model.

[3] http://www.uebeq.com

In addition, because configuration often requires specific refinement of the software, its rigidity, fragility, opacity and immobility [1] is emphasized, hence compelling its re-engineering.

Consequently, the goal of the project is to facilitate the software configuration by business experts and consultants while improving the configurability and adpatability of the software. For that, it is prescribed an approach combining:

- ontologies, that allows the configuration of the business concept according to the enterprise's business, department or individual employee;
- rules, that allows the specification of business rules/constraints upon the ontological concepts, which goes beyond the expressivity of the ontological constraints.
- DSL (Domain-Specific Language), developed with the purpose to facilitate the business experts/consultants to configure the system with minimal or no technical skills.

2 Proposal

The legacy software (left-hand side of Fig. 1) adopts a client-server architectural style, whose backend (i.e. the server part) evolved into a large monolithic , developed in a myriad of frameworks (e.g. ASP, ASP.Net) and programming languages (e.g. C#, VB.Net). The backend is responsible for the data access (from/to relational database), business logic and server-side generation of the Rich Internet Application client-side UI pages (in HTML and JS).

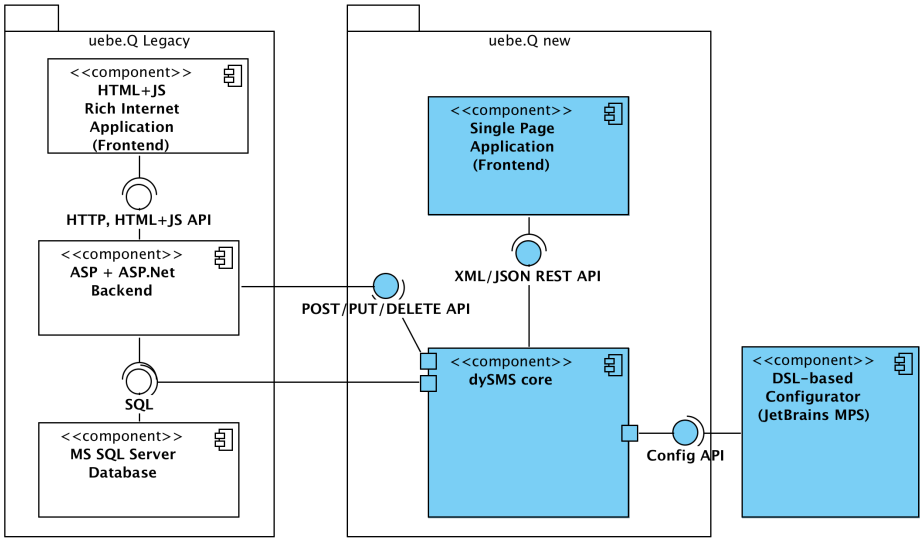

Fig. 1. Component-based diagram of the new uebe.Q and configurator.

While the business logic captured by the backend is a relevant asset, it suffers from rigidity, fragility, opacity and immobility [1], frustrating a conservative and traditional evolution. In that sense, the system has been re-engineered, exploiting the functionalities (i.e. business logic) of the legacy system while allowing the creation of new ones and their definition at configuration time. The added components (at the right-hand side of Fig. 1) include the dySMS Core, the SPA frontend and the DSL-based configurator.

The dySMS Core component reads data directly from the legacy database, but the original business logic might be needed to update, delete or create data.

The DSL-based Configurator allows the business consultant to define the domain-specific ontology (model) and respective (business) rules. The DSL has been developed in the JetBrains Meta Programming System [4] which provides a projectional editor that drives the configurator tasks according to the DSL-captured structure and semantics (Fig. 3).

The semantics defined through the ontology and rules follows a hierarchical policy, in which the semantics defined at a hierarchy level binds to the semantics defined in top hierarchy levels, promoting organizational reasoning consistency.

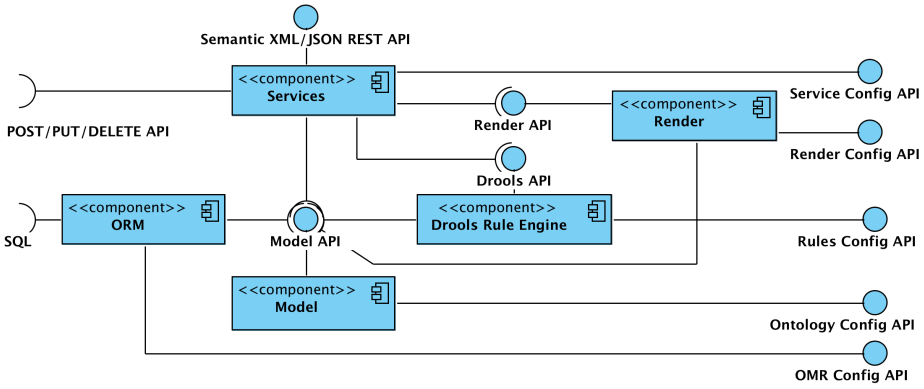

Fig. 2. Detailed component-based diagram of the new uebe.Q.

Fig. 2 depicts the inner architecture of the backend, including the identification of the parts/responsibilities and the details of the configuration interfaces. The system has the following components:

– Model, is the set of classes and respective instances that are managed by the system. The classes, associations and constraints corresponds to the ontology(ies) defined by the consultant using the DSL-based Configurator (cf. Fig. 3). A reflection-based approach generates on-the-fly the configuration-time-defined classes, allowing the system to operate with whatever classes structure is required.

[4] https://www.jetbrains.com/mps/

– ORM, which is responsible for the data access of the relational database and the instantiation of the classes. The mapping between the ontology and the database schema is defined by the business consultant using the DSL-based configurator (cf. Fig. 3) complemented by an ontology mapping information integration decision support system (cf. [2]).
– Drools (https://www.drools.org) is a closed world assumption rule engine that reasons upon the model/ontology using a Closed World Assumption (CWA).
– Render is responsible for the projection and serialization of the data according to the authorization, projection and serialization rules, also configured.
– Services is the set of semantic REST web services that are responsible for the availability of the new functionalities. The component delegates the responsibilities in the other components as suggested in the diagram.

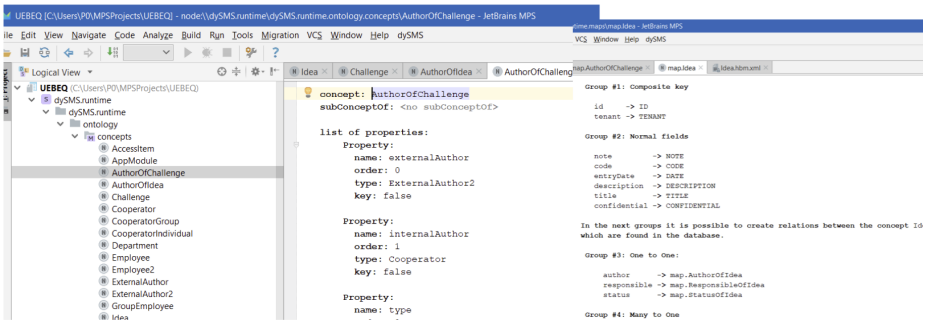

Fig. 3. Screen shots of the DSL-based configurator application.

3 Summary

The combination of ontology, rules and the CWA reasoning provided by Drools allows adopting the same configuration/specification approach to different dimensions of the system, such as business rules, ORM, authorization and rendering. Running efforts are focused in adopting the approach in further customers and its adoption to other software.

References

1. Martin, R. C.: Designing Object Oriented Applications using UML. 2d. edn. Prentice Hall (2000)
2. Gouveia, A., Silva, N., Martins, P.: A Rule-Based DSS for Transforming Automatically-Generated Alignments into Information Integration Alignments. In: 19th Int. Conf. on Inf. Integration and Web-Based Applications & Services, pp. 181–87. Salzburg, Austria (2017)