Model-Based Generation of Enterprise Information Systems

Kai Adam[1], Lukas Netz[1], Simon Varga[1], Judith Michael[1], Bernhard Rumpe[1], Patricia Heuser[2] and Peter Letmathe[2]

Abstract:

Thick clients of client/server-information systems include increasingly more logic which leads to several challenges in the development process: Resulting from the separate development of front- and backend, the risk for inconsistencies between components on the one hand, and communication overhead between developers on the other hand are high. We present an approach which helps to overcome these challenges by using model-driven engineering for the development of data-intensive enterprise information systems. WebDEx was developed as a generator for the creation of such systems. It uses UML/P inspired modelling languages, as models (1) build the base for communication among project members and (2) are used as input for the code generator which ensures consistency by construction. This work relies on an infrastructure created by the language workbench and code generation framework MontiCore. Moreover, this paper presents the practical application of this approach for the agile development of a multi user, adaptable web-application to be used by more than 400 chairs of RWTH Aachen University.

Keywords: Data-Intensive Enterprise Information Systems · Model-Based Software Engineering · Model-Driven Information System Development · Multi-User Web Applications · WebDEx · Agile Development

1 Introduction

In the development of web-applications with classical thin clients, logic is defined on the server-side. In modern architectures, thick clients, also known as smart clients, assign a part of the functionality to the client-side. This type of software design is getting more in common in modern architectures [JW06]. Hence frontend and backend are implemented separately and often use different programming languages. Developers of both parts have to communicate intensively and have to adjust their implementation to the corresponding other side. Consequently, this increases the risk for communication overhead and inconsistencies.

Abstract models have proven to be a good approach to face these challenges. They can be used as a foundation to all parts of the implementation. These models provide a viable input for generators to create source code, which is consistent among various parts of the project

[1] RWTH Aachen University, Software Engineering, Germany {nachname}@se-rwth.de
[2] RWTH Aachen University, Controlling, Germany {nachname}@controlling.rwth-aachen.de

(consistency by construction). Domain experts with modelling skills are able to develop and customize these models within their respective domain without being involved into details of the implementation or other domains (similar to [HMM18]). Models build the base for communication among project members. Moreover, changes of the models are easily passed on to the source code and decrease development time. This provides advantages for each of the three intended user groups: *The developer* needs only specific knowledge about the implementation e. g. of types, but not about the involved domains. *Domain experts* with modelling skills develop the models (data structure and GUI design) within their scope, and need no knowledge about implementation details. *The end user* benefits from fast handling of changes, easy adaptability to new guidelines and quick implementation of feature requests.

This work is based on the model-driven software engineering (MDSE) experiences of the SE group of RWTH Aachen university and the developed MontiCore (MC) language workbench and code generation framework [KRV10]. Models, created with UML/P [Ru16] inspired modelling languages, are used as input for this framework. The presented approach for the development of enterprise information systems (EIS) is grounded in preliminary work of the research group, e. g. several theses and the latest developed MontiCore Data Explorer (MontiDEx) code generator [MRR15]. MontiDEx processes models to generate data-centric applications in Java and Java Swing. The current approach includes the movement to a different technology stack (Angular as a framework for client applications). Thus, a new EIS generator called WebDEx was developed. There exist several approaches for the generation of web applications using different platforms, e.g., Bernardi et al. [Be12] integrate three meta-models based on a declarative language. Given the page limit, we refer the readers for further discussion of related work to `http://www.se-rwth.de` (publications and phdtheses).

The presented approach has a high proportion of generated code and shows its' practical application with a case study creating a data-intensive EIS within the MaCoCo project[3]. An important goal of the project was the agile development and adaptability of this multi-user web-application. Thus, lead users are involved actively in the development process, new features are delivered quickly and frequently and the project team reacts fast to changes with a focus on improving the quality of the EIS. MaCoCo will be used for financial and staff management of more than 400 chairs of the university. Moreover, the findings from building generic abstractions make our approach reuseable for other EIS development projects.

The paper is structured as follows: Section 2 specifies the general concept of the approach and its' main advantages. Section 3 presents the first practical realization of our approach in the MaCoCo project. The last section reviews the current progress and highlights further goals and next steps for our approach.

[3]The MaCoCo project is funded by the RWTH Aachen University and jointly realized by the chairs of Controlling and Software Engineering.

2 MDSE for data-intensive EIS

Our approach for the model-based generation of data-intensive EIS (Figure 1) consists of three major components: (1) A set of *models* describing the software that will be generated as input, (2) a powerful *generator*, including a set of parsers, capable of interpreting given models and (3) the *target*, where the generated sources will be built in (in this case realized as web-applications).

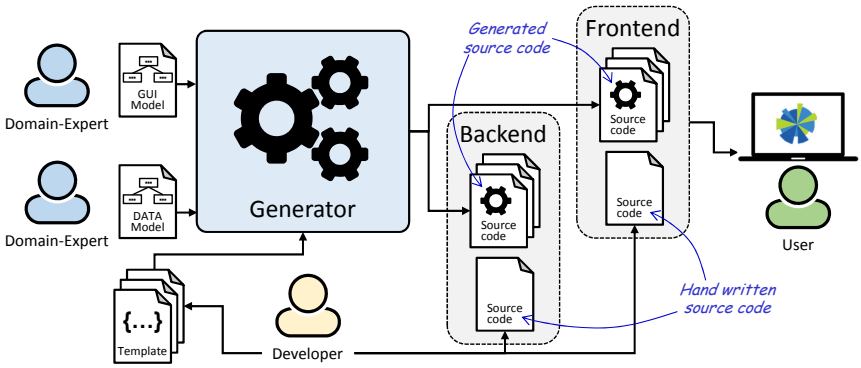

Fig. 1: Approach for the model-based generation of data-intensive EIS with the WebDEx generator

(1) Domain experts provide models in corresponding textual domain specific languages (DSLs): A GUI-designer provides a GUI-model (based on MontiViz [Re16]), whereas different domain-experts provide data-models, e. g. class diagrams (created with UML/P [Ru16] inspired modelling languages). These models describe different aspects and components of the software.

(2) The generator interprets the models with parsers (created with MontiCore[4]). In a next step, the generator detects conflicts between the given models, e. g. names are assigned twice, and applies standards which are defined at a global level, e. g. getters and setters for each data class or default parameters. Once the abstract representations of the models are processed, the generator uses platform specific *templates* to generate source code. Such a template is a blueprint for code fragments in a certain programming language, like method or attribute definitions (e. g. how to write a `toString()` method in Java code). They can be exchanged to generate source code in different programming languages, while using the same set of models.

(3) Depending on it's configuration, the generator will create code for both front- and backend. The generated code is easy to read and interpret, can be easily extended to include hand written code, and reacts well to model changes by domain experts.

[4]MontiCore provides parsers for textual DSLs if provided with a corresponding grammar. See `http://www.monticore.de/` for more information.

To sum up, the main advantages of using the generator are: it works iteratively and deals thus well with changes, it is responsive to existing hand written code and already provides extension points to enable the enhancement of the generated functionality by hand.

3 Practical Realization: MaCoCo

In academia research projects' *financial management* is becoming increasingly demanding: Different funding authorities, a variety of funding schemes and especially diverse requirements for accounting make it a challenging field for researchers and administration. Similar challenges occur for *staff management*, where different sorts of contracts and salary schemes, official restrictions for employments and changing assignments to projects keep administration busy. The more *projects* a chair has, the more researchers are hired and the more important it is to keep an overview.

Out of this need, the chairs for Controlling and Software Engineering started the MaCoCo (Management Cockpit for Controlling) project[5], which realizes a data-intensive EIS as web application. Supported features are extended financial and staff management as well as course administration for professors. MaCoCo is intended to replace and standardize existing management processes and is planned to be used by up to 400 chairs of RWTH Aachen University.

In a first realization step, a handwritten prototype has been realized to outline the project's scope. Based on this source code a generator was created to replace the handwritten parts step-by-step with generated code [Gr06]. Whereas it may seem quite unusual to develop a project like this, this first realization step was very useful to evaluate the initial concept and get more into the domain. Anyhow, here is no need to follow this way of realization in other projects: The generator can now be used to create a new application, without preceding handwritten implementations.

As readers can imagine, each chair has different requirements for such a system and they can change over time due to external influences. This means *agile development* for quick implementation of requests is a must. Using a model driven approach in combination with the generator makes it possible to react quickly to changes in the data model and still produce a consistent product. Changes of data types of existing attributes or adding new attributes in the model will result in a corresponding change in the generated application. Currently nearly two thirds of the dynamic parts of the application are generated. Runtime environments and services are currently hand written, but could be easily generated as well.

[5] See `https://git.rwth-aachen.de/macoco/extern/wikis/home` for further details.

4 Conclusion

To sum up the main contribution of this paper: The presented approach makes it possible to generate large parts of an EIS and to develop such systems in an agile manner. As a proof of concept we presented the practical application of the approach in a case study. It is possible to use multiple models as an input and to generate source code in two different programming languages (TypeScript, Java). The *most challenging aspect* was building abstractions of the existing hand written source code fragments. Generic parts had to be identified in order to efficiently create templates. Nevertheless, once these generic abstractions exist, they are reusable for other EIS development projects.

Up to now visualizations and SQL statements are not generated. Thus, *next steps* include the development of a generator for visualization models and the generation of SQL statements related with the visualization. Furthermore, we are interested in using additional models, e. g., state charts and sequence diagrams, as generator input to improve program logics.

References

[Be12] Bernardi, M. L.; Cimitile, M.; Di Lucca, G. A.; Maggi, F. M.: M3D: A Tool for the Model Driven Development of Web Applications. In: Proceedings of the Twelfth International Workshop on Web Information and Data Management. WIDM '12, ACM, Maui, Hawaii, USA, pp. 73–80, 2012.

[Gr06] Grönniger, H.; Krahn, H.; Rumpe, B.; Schindler, M.: Integration von Modellen in einen codebasierten Softwareentwicklungsprozess. In: Modellierung 2006 Conference. Vol. 82. LNI, pp. 67–81, 2006.

[HMM18] Hernandez-Mendez, A.; Michel, F.; Matthes, F.: A Practice-Proven Reference Architecture for Model-Based Collaborative Information Systems. Enterprise Modelling and Information Systems Architectures 13/, pp. 262–273, 2018.

[JW06] John, S.; Wi-Mei, M. H.: A proposed framework for an effective integration of supporting environments for smart client application development. In: Int. Conference on Computing Informatics. Pp. 1–6, 2006.

[KRV10] Krahn, H.; Rumpe, B.; Völkel, S.: MontiCore: a Framework for Compositional Development of Domain Specific Languages. International Journal on Software Tools for Technology Transfer (STTT) 12/5, pp. 353–372, 2010.

[MRR15] Mir Seyed Nazari, P.; Roth, A.; Rumpe, B.: Mixed Generative and Handcoded Development of Adaptable Data-centric Business Applications. In: Domain-Specific Modeling Workshop (DSM'15). ACM, pp. 43–44, 2015.

[Re16] Reiß, D.: Modellgetriebene generative Entwicklung von Web-Informationssystemen. Shaker Verlag, 2016.

[Ru16] Rumpe, B.: Modeling with UML: Language, Concepts, Methods. Springer International, 2016.