

Relational Model Driven Application Design

Michael Guckert, René Gerlach

Fachhochschule Gießen-Friedberg
Wilhelm-Leuschner-Straße 13
61169 Friedberg/Hessen
Michael.Guckert@mnd.fh-friedberg.de
Rene.Gerlach@transmit.de

Abstract: A closer look at typical information systems shows that relatively simple routines often contribute significantly to the overall expenses of the software development process. Moreover an empirical study examining the commercial off-the-shelf solution eVMS for the administration of contracts in the public transport business revealed that structurally similar simple code outweighed (measured in lines of code and the number of online dialogs) complex code that implements complicated business processes. Exploiting structural database information enriched with additional application specific metadata allows automation of routine tasks thus reducing development expenses and additionally promising higher product quality. RMDAD transforms relational database metadata into a generic object model. At runtime the RMDAD-interpreter exploits this object model to dynamically generate application code on the fly.

1 Motivation

Due to the competitive orientation of the German public transport market the process of managing contracts between the representatives of public administration and the typically private transport providers becomes increasingly complex. The use of an integrated information system that supports the administration of the complete life cycle of such a contract improves the quality of the transport supply significantly. The electronic contract management system eVMS was originally built for that purpose in a project for a public transport organization [Fr05]. Meanwhile eVMS evolved into an off-the-shelf product and is now used by three of the largest public transportation organizations in Germany.

Software complexity can simply be defined as the degree to which a system or component has a design or implementation that is difficult to understand or verify [IEEE]. A more elaborate model [FP00] differentiates between four types of complexity: algorithmic complexity, problem complexity, structural and cognitive complexity. Following this definition, code for the maintenance of master data typically has a low complexity in all four of these dimensions. It mainly consists of the implementation of the CRUD processes (Create, Read, Update and Delete). Moreover code that implements this functionality always is more or less structurally equivalent.

Code analysis of original eVMS implementation showed that 50% of the overall code belongs to the class of simple master data maintenance [Ge07]. We used lines of code and the number of online dialogs as measures for the amount of programming work, testing and integration tasks. Experience shows that such routine tasks are typically more error prone than complex code because they are intellectually less challenging and reduce the programmer's attentiveness. Therefore the main ambitions of RMDAD are reducing programming efforts and increasing product quality by substituting manually coded routine dialogs by automated components wherever that is possible.

2 RMDAD – Idea and Implementation

RMDAD uses standard database metadata enriched with additional application specific information and the current user input to create and configure the necessary Struts components at runtime. General configuration parameters, the layout and the language are additional controls for that code generation. All the same RMDAD is a 100% Struts program that can be seamlessly integrated into existing Struts applications. RMDAD uses an interpreter so that no build process is necessary. RMDAD simplifies the time-consuming task of developing web applications within the Struts framework (editing control files, writing JSP, coding actions forms etc.) significantly. Once layout characteristics (fonts, colors, etc.) are defined in CSS files it suffices to create the tables in the database to generate the complete CRUD dialogs for the web application.

The RMDAD database metadata object model is an abstraction of a generic database catalog. Among other things it contains information about the table structures, the columns with their domains, keys and relations.

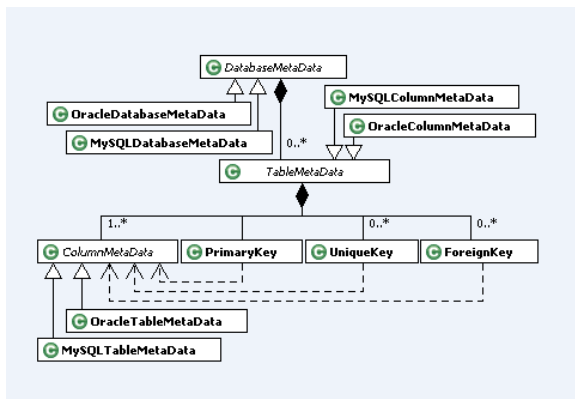


Fig. 1: Metadata Model

The metadata model is used to dynamically create the SQL statements for database access at runtime. Further relational information is used within the GUI to create a convenient user friendly interface, e.g. Foreign Key information is evaluated to populate drop down boxes.

The abstractions of the database structure and of the database content described above ensure that RMDAD can easily be adapted to different relational database systems.

3 RMDAD – Conclusions

Up to now RMDAD can automate relatively simple master data maintenance functionality. The more complex business logic of eVMS is implemented using conventional Struts code. Of course RMDAD reduces individuality and standardizes processes. But this is typically not an issue in master data maintenance dialogs. Application components that actually do require higher individuality can still be implemented with conventional Struts. The use of CSS and the flexibility implemented with the help of configuration parameters allow simple adaption of the system's look and feel according to specific requirements.

Changes or improvements in globally used functionality can easily be implemented by adapting the interpreter. That means that no editing of JSP files is necessary for global adaption of the user interface as all the elements of the forms are generated dynamically by the interpreter. Testing effort is reduced significantly because instead of testing many separate dialogs we can now concentrate on the interpreter and the correct definition of the database structures.

The analysis and design phase of the initial eVMS project produced Entity-Relationship-diagrams that were transformed into relational models and database structures. During the ongoing development process the original ER-diagrams were not synchronized with the evolving application structures. Therefore we searched for an MDSD approach that allows a high degree of automation in the given Struts environment without having to (re-)produce design artifacts. Therefore we abandoned the idea of representing the basic structures of the application in a more state of the art form like UML that would suit the object oriented development process better. We think that this not an unusual situation in the industry and is in fact the typical case where RMDAD can be used very effectively.

4 Literature

- [FP00] Fenton, N. E.; Pfleeger, S. L.: *Software Metrics - A Rigorous and Practical Approach*, Thomson Learning, Second Edition, 2000.
- [Fr05] Frey, V.: *Elektronisches Vertragsmanagement im Rhein-Main-Verkehrsverbund*, in: Der Nahverkehr 12/2005, Düsseldorf, 2005.
- [Ge07] Gerlach, R.: *Datengetriebene Softwareentwicklung*, Diploma thesis, University of Applied Sciences Gießen-Friedberg, Friedberg 2007
- [IEEE] IEEE Std. 610.12-1990: *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std. 610.12-1990.