REUSABILITY IN PATIENT REGISTRIES
Implementation of a Generic Extensible Web-based Patient Registry System

Eric Tröger, Elena Prokofyeva, Robert Wilke
Biomedical Engineering Lab, Institute for Ophthalmic Research, Centre for Ophthalmology, University of Tübingen
Paul-Ehrlich-Strasse 17, Tübingen, Germany

Eberhart Zrenner
Institute for Ophthalmic Research, Centre for Ophthalmology, University of Tübingen
Schleichstrasse 12-16, Tübingen, Germany

Keywords: Patient registry, patient database, generic database, Operational Data Model, ODM, XSLT, ophthalmology.

Abstract: Usually patient registry systems are developed in very specific scientific contexts where the reusability is considered a topic of minor concern. To improve reusability as well as to achieve better separation between the data definition step and the technical implementation of the patient registry, a generic approach has been chosen to automatically generate application components based on the data definition. This is done by using the ODM standard format as meta-model and applying XSL transformations to create and configure system components. Components required among several patient registries are implemented once and reused.

1 INTRODUCTION

To allow scientific analyses of large amounts of patient datasets as well as to assemble populations of patient datasets for recruitment for upcoming clinical trials it is of great importance to have a robust and easily accessible patient registry system.

Usually such a system is built focusing on a very specific scientific purpose - and therefore has limited usefulness for other applications – or patient data is gathered from paper files or operative systems on demand. Problems regarding technological as well as organisational aspects are usually appearing frequently among such systems.

In order to enhance reusability, we are pursuing a generic approach for our patient registry system, enabling free specifications of data definitions by using a standardised meta-model. This approach not only allows to automatically create user interfaces like Web-based data entry forms, it also enables interconnectivity among multiple systems generated by this technique by matching the models used for the generation-processes. It also ensures the reusability of a highly robust technological framework.

2 CONSIDERATIONS

In medical contexts scientific expertise usually has a crucial influence on data structure design. Although data definition is one of the most important steps while developing a patient registry, it can also become very time consuming and may unacceptably delay further development steps.

Especially within larger consortiums where differing scientific interests are present, it can be difficult to reach a point of mutual consent. This e.g. held true for the common ophthalmic patient registry system developed within the scope of the European Integrated Project EVI-GENORET. The system covers about 400 data fields per patient, and while the overall system has already been serviceable from a technical point of view, field definition changes needed to be made even after patient data already had been entered due to continuously developing scientific requirements.

As a result our current approach in building patient registry systems aims at a strict separation of the data definition process and the technical implementation of the system. While core components are taken from a common set of program modules, data definition specific parts are mainly generated during compile time.

3 IMPLEMENTATION

3.1 Technological Infrastructure

Since our previously developed patient registry systems are J2EE Web-applications, the generic approach also targets this kind of application. Following common best practices, the framework is structured as a 3-tier application utilising widely used libraries for implementing the different tiers. Hibernate 3 is used as object relational mapping layer to allow for database abstraction. Struts 2 is used as Web-application framework realising the MVC pattern, whereas Freemarker is used as template engine implementing the View to render HTML-pages. Furthermore certain already implemented components from previous systems are reused, e.g. the auditing component logging user activity on the database (Section 3.5.2).

3.2 Generic Approach

It is essential to point out that the target system is clearly defined as patient registry system and therefore allows for restrictions on what components are to be generated and to which extent. Our efforts were not directed towards generating arbitrary configurable database systems, but rather focused on this special kind of database system. This also leads to implications e.g. regarding data security and protection components.

While our approach cannot directly be defined as such, it still can be considered as being related to Model-Driven Software Development (MDSD) in terms of building a software system based on a model, whereas the model itself first and foremost contains data structure definitions.

3.3 Meta-Model

A common meta-model was needed as basis for the generic platform. After searching for an appropriate meta-model as well as considering defining a new meta-model, the CDISC (Clinical Data Interchange Standards Consortium) Operational Data Model (ODM) has been chosen.

ODM is a common standard for interchange of clinical data, especially in clinical trials systems. ODM is an XML-dialect, which was being supplemented by Vendor Extensions to allow for the usage as platform-specific model. The ODM-file not only contains definitions of data structures, but also validation rules, user interface configuration and access control rules that allow to configure database access rights as will be described in the following sections

3.4 Compiling an Application

After defining an ODM file a patient registry system can be built automatically. Since about half of the target application consists of reusable components (Section 3.5) this step can also be thought of as configuring the target system.

3.4.1 Principle

The system is built by Maven 2. The ODM file is being transformed by multiple XSL files using an XSLT Maven plug-in, whereas different components are generated. While some XSL files only produce one target file, others produce many at once, depending on the type of the transformation. There is a difference if only one configuration file is to be generated, or multiple classes representing domain entities. Internationalisation is completely supported throughout all generated and non-generated components and within the ODM file.

3.4.2 Interfaces for the Datasets

The ODM file is structured into StudyEventDef-Elements (XML tags), whereas each of them can reference a FormDef-Element. The StudyEventDef-Elements are use to model specific groups of datasets belonging together. In our sample patient registry the StudyEventDef “Monogentic Retinal Dystrophy” encapsulates FormDefs representing sub datasets for this kind of disease like “Clinical Examination” or “Technical Examination”.

The FormDef-Elements are used to generate Java interfaces representing the corresponding datasets, e.g. for “Clinical Examination”. For FormDef-Elements references to ItemDef-Elements are modelled, and for each ItemDef an identifier and the data type are declared. This information is taken to generate corresponding getter and setter methods for the Java interfaces representing the datasets.

3.4.3 Object Relational Mapping

To configure the database abstraction layer, Hibernate 3 specific domain classes are generated that implement the interfaces described in Section

3.4.2. Those classes are automatically extended with Hibernate Annotations containing database mapping instructions of which most are dependent on the ODM content.

Furthermore service classes are generated that allow for database operations as database access objects (DAOs).

3.4.4 Web-Forms

The target patient registry system is Web-based and therefore requires HTML-forms to show and enter data. Since Freemarker is used to generate HTML output, Freemarker templates (FTL) are generated at compile time. One FTL represents one dataset defined as FormDef in the ODM file, and therefore will be associated with one specific Hibernate domain object (Section 3.4.3) at runtime.

The ODM file has been extended by Vendor Extensions to accommodate additional tags describing a basic user interface layout for each dataset. This was done as intended by CDISC by adding custom XML Schemata (XSDs) referenced by the ODM file. This layout information is not only used to structure Web-forms, but also for reports (Section 3.4.7). It enables visual grouping of elements.

3.4.5 User Input Validation

To facilitate the definition of validation rules within the ODM file Vendor Extension tags have been added to allow for data entry restrictions like regular expressions. Range checks are also possible for numerical data types.

Apart from manually defined validation rules, data type dependent rules are added at compile time to e.g. automatically check against date-format patterns when data is entered. All validation rules are converted into an application specific format and applied at client side (JavaScript), as well as at server side (Struts 2 Interceptor).

3.4.6 Restrictions on Data Collection

The ODM standard provides Collection-Exception-Conditions that can be set for each item referenced within FormDef-Elements. Those conditions were used to set user input restrictions by defining special XPath expressions.

An example expression may look like:

```
../ItemDef[@OID='CLINICALEXAMINATION
.ANTERIORSEGMENT.IRIS.ABNORMAL.OS'][
@Value='OTHER']
```

Each item referencing this condition as Collection-Exception-Condition is only to be collected if the Item with the given ODM identifier has the value "OTHER". This ODM compatible expression is converted into an application-specific rule at compile time. These rules are not only used to show/fade-out user input elements on Web-pages while entering data depending on previously entered data (JavaScript), but also to validate the data before saving it in the database (Struts 2 Interceptor).

3.4.7 Reports

For each dataset a printable report can be generated when using the target application. The reports are compiled and filled by JasperReports, which needs JRXML files as templates. At compile time JRXML files are generated using XSTL, whereas the structure of the reports is mainly defined by the user interface layout defined in the ODM file.

After the JRXML files have been generated, they are compiled by the Maven Jasper plug-in into Jasper files to enhance performance at runtime.

3.4.8 Option-Lists for Data Fields

Many data field values are restricted to specific options which can be defined in the ODM file as CodeLists. The values defined in the CodeLists are parsed and used at runtime to fill select-boxes on Web-pages or to validate entered data before saving it.

Special CodeLists can be marked as persisted within the database, which is appropriate for large sets of available options. This was true for long custom diagnosis lists or ICD10 lists in our sample application. As additional feature options stored within the database can be structured as a tree. A special AJAX-based user interface component has been developed to handle tree-structured option lists.

3.4.9 Additional Generated Components

The ODM file allows the configuration of the target patient registry in much more depth than has already been described.

For example file upload components can be specified, causing special tables holding binary content to be generated. Additionally AJAX-based file upload user interface components are then integrated into the Web-forms.

Web-based search masks are pre-configured according to ODM content allowing restrictions for the searchable values of the AJAX-based query builder and backend search engine. So if only a predefined range of values was allowed for a numerical field or a specific set of options is defined for a field, this will be respected when dynamically building the user interface.

Further generated components like an additional importer module to integrate data from external databases can be compiled on demand. The importer also includes all validation checks that are performed when manually entering data. Further data integrity checks are performed by validating the data to be imported (XML format) against an automatically generated XML Schema.

3.5 Reusable Components

Apart from the generated portion of the target application, many components are shared among all patient registries. Some of them are being addressed in the following sections.

3.5.1 Application Framework

The basic infrastructure of the application covering HTTP-request handling and HTTP-response filling (Web-pages, reports, binary streams), as well as database connectivity, logging, HTML-rendering fragments, search routines or the validation framework are common for all generated patient registries.

The framework can be augmented and extended separately, providing features for all patient registry systems to be developed in the future.

3.5.2 Data Protection and Security

To provide regulation compliant data protection and security mechanisms that are required for medical databases, the authentication system is constructed as non-generic component. ACEGI is used as extensible security framework.

Security rules and roles can be defined in the ODM file, both on dataset and attribute level. Typical rights include read, write, edit, search, export and are evaluated throughout different layers of the application.

An auditing mechanism logging all changes made to patient datasets on attribute level guarantees complete traceability. This also covers non-manual database access like initiated by the importer.

3.5.3 User Management

User data is stored within the database at the moment, whereas different user management techniques may be implemented in the future. Users can register themselves using a registry form. Afterwards an appointed administrator can approve or reject pending registrations for his clinical centre.

The user registration is handled via email. Configurable templates (Freemarker) are used to personalise the email content.

4 CONCLUSIONS

The generic system was successfully used in Tuebingen to construct a patient registry named Ophthabase. Although created for ophthalmology, the generic concept allows the creation of almost any kind of patient registry, either as standalone system or as basis for a distributed patient registry within a scientific consortium.

The generated patient registry can easily be modified or extended by adjusting the ODM-file, which saves much time especially in situations where the data structure definition is still under review. The generated system is less error-prone during system development, since data modelling is strongly separated from technological concerns.

The generic concept will be extended in the future to further augment the patient registry systems, e.g. by adding more sophisticated search routines, analytic functionality, miscellaneous export modules, and optional pedigrees.

Multi-centre data interchange will be possible based on Web Services using a specialised mediator-server able to match ODM-models of different patient registries. Patient registry systems will be able to communicate directly after mediation.

By removing the Vendor Extensions from the ODM-file it can also be used to configure clinical trial systems. To amend the definition of the ODM-file a specialised visual editor may be developed. It is also planned to provide a tight integration into isTUmas, the study management system developed at the University of Tuebingen (Strasser el al., 2008).

REFERENCES

Clinical Data Interchange Standards Consortium, http://cdisc.org, accessed 15/07/2008

European Vision Institute – GENORET, http://www.evi-genoret.org, accessed 15/07/2008

Strasser *et al.,* 2008. *An integrated System for Workflow and Data Management in Clinical Trials*, ARVO 2008 Annual Meeting. Fort Lauderdale, April 27, 2008 – May 01, 2008.