# A Meta Model-Based Web Framework for Domain Independent Data Acquisition

Dominic Girardi and Johannes Dirnberger
Research Unit Medical Informatics
RISC Software GmbH
Hagenberg, Austria
Email: dominic.girardi@risc.uni-linz.ac.at
Email: johannes.dirnberger@risc.uni-linz.ac.at

Johannes Trenkler
Institute for Radiology
Landesnervenklinik Wagner Jauregg
Linz, Austria
Email: johannes.trenkler@gespag.at

***Abstract*—We present a generic, web-based data acquisition system that is based on a domain-independent meta data model and which is able to collect, store and manage data of almost arbitrary structure. Due to the use of abstract meta data models, completely generic applications can be built. The additional level of abstraction guarantees the independence of database structure and source code from the actual domain of application and allows to create software systems that can be customized for a certain application without changing any internals of the system. So, domain experts and researchers are able to create, run, and adapt their own web interface for data acquisition without depending on external IT experts. We demonstrate our approach on a registry for intracranial aneurysms.**

***Keywords*—Meta-Modelling; Web-based Data Acquistion; Generic Data Acquisition Systems.**

## I. INTRODUCTION

Data analysis and data mining are well established and indispensable tools in medical research. But they cannot be seen as isolated steps. They are included into a process of knowledge discovery (KD). Cios et al. [1] define this process as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Besides Cios et al., several other definitions and descriptions of the knowledge discovery process (KDP) can be found in [2], [3], and [4]. They all describe KD as a process usually beginning with data selection, data cleaning and transformation, followed by data mining, finalized by interpretation and presentation.

However, one particular aspect is often neglected: data acquisition. The reason for this neglect is the fact that, usually, data, which is produced anyway in business processes, genetic experiments, etc. is analyzed. But, especially in scientific medical research, it is often necessary to acquire the data of need. Data stored in hospital information systems (HIS) is hardly suitable for scientific research because it is often semi-structured, textual data [5] or contains data mostly for billing and documentation purposes [6]. Although data mining has already been performed directly on HIS, those results are less scientifically applicable than for management purposes [7], [8].

Since medical data structures are usually non-trivial, a professional data acquisition system is needed to acquire and store the data. Since every field of study requires its own individual data structure, these data acquisition systems are hardly reusable and need to be developed individually for each new domain. This is elaborate and expensive and often makes study authors prefer suboptimal data storage solutions (i.e., MS Excel sheets). In order to overcome these drawbacks, we have developed a generic, web-based data acquisition system which is based upon an abstract meta data-model. Several existing approaches try tackling this goal. Frameworks like *Hibernate* [9] or new programming languages like *Ruby* [10] *on Rails* [11] have their focus on supporting the programmer rather than the end user. Our approach is focused on providing a system that can be easily adjusted by the end-user directly without information technology (IT) knowledge.

The application of a meta data-model allows the creation of an abstract domain-independent system that is then domain-customized by the domain expert himself. Intuitive user interfaces allow the medical domain expert (who is probably not an IT expert) to define his data structures of interest. The rest of the system (UI forms, overview tables, search masks, etc.) is automatically created based on the stored meta-data. So, the user is now able to set up and maintain its own medical study without dependence on an external IT contractor.

In Section II, we provide an overview over related publications. Section III contains a detailed desription of the generic meta model and the application itself. In Section IV, we present our results on the example of a disease register, which was realized using the system in the Landesnervenklinik Wagner-Jauregg Linz. Section V contains our conclusion and an outlook for further research.

## II. RELATED RESEARCH

Our literature review in scientific sources did not yield any scientifically published system that is directly comparable to our approach. However, in the field of medical data recording there are a number of systems which offer eletronic case report forms (eCRF). Franklin el al. [12] compare three of the most popular electronic data capture (EDC) systems Catalyst Web Tools, OpenClinica and REDCap [13]. These systems are very complex and offer additional features for study management and planning. But still, they are limited to clinical data acquisition and not as generic as our system. A more generic approach can be found in Zavaliy et al. [14]. The one page position paper describes the basic concept of an ontology-based data acquisition system for electronic medical record data. A very simple ontology is used, which contains four

 ISBN: 978-1-61208-283-7 

concepts: Person, Hospital, Diagnosis and Medication. They point out, that the main reason for using an ontology-based approach is the need for adaptive data structures. This work is closely related to our work. However, their paper contains no information on how data can be entered into the system, neither is there information about the system architecture or semantic data checking. There is also no information given on how adaptable their ontology is and if those four main concepts can be replaced or not. Despite the fact that they follow a very similar basic idea, our system seems to be more extensive and matured.

Apart from these, at least strongly customizable but not absolutely, generic systems some domain specific web-based data acquisition systems can be found, such as Kodama et al. [15], for collecting wind data.

## III. METHODS

### A. Meta Model

*1) Motivation:* Conventional data acquisition systems are based on a descriptive data model, which directly reflects the domain of application. Since the whole application is based on the data model, there is a strong semantic dependency from the domain of application through the data model to all other aspects of the application (logical layer, user interfaces, data exchange interfaces). Due to this dependency, changing requirements (additional attributes, changing tables, adding relations, etc.) cause changes throughout the whole application, which usually needs to be performed by IT experts. Furthermore, this dependency strongly limits the reusability, since different domains of applications cause completely different data models.

The usage of meta data-models leverages this dependency. The domain-specific data structures do not define the data structure of the application but are stored into the meta data-structure. So, the generic meta data-structure always remains the same, regardless of the actual field of application. The stored meta data allows the automatic creation of GUI elements, so changes to the domain-specific data structures are automatically propagated throughout the application.

*2) Meta ER Model:* Our meta model is based on Peter Chen's entity-relationship (ER) model [16]. It is basically an ER model which is able to store another ER model and its data. Figure 1 shows the core elements of our model. The elements on the left hand side are primarily used to define the data structure, while the elements on the right contain the actual data. For further technical details the reader is referred to [17].

*3) Consequences:* Due to the abstraction of the whole system from the domain of application, a lot of advantages arise for both the user and the developer. The user is now able to set up and maintain the system without dependence on an IT contractor. The changes he makes to the domain data model have immediate effects on the user interfaces. By using meta models, we were able to create a system that is highly adaptable and reusable. However, developing generic systems also means dealing with an additional level of abstraction. Especially, when it comes to database queries, meta data-models can be very challenging. Data, which can be easily queried from a conventional database, needs to be joined from multiple tables in the meta model. This is not only a performance issue; it also complicates the query structure.

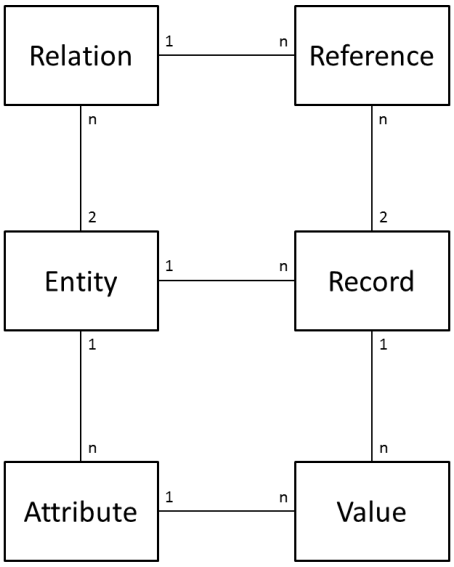


Fig. 1. ER diagram of the meta data-model [17]

*4) Relation to OWL Ontologies:* Ontologies are content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge [18]. So, the presented meta data-model can be seen as an ontology storage and the system itself as a ontology-based data acquisition system. Consequently, the question arises, why the ontology definition languages OWL or RFD were not used for this system. The main reason is the paradigm mismatch between the Open World Asumption of (OWL) ontologies and the Closed World Asumption of relational data models. In OWL anything, that is not forbitten is allowed. So, for example, a record (individual) can be instance of more than one class, which does not make sense for medical data acquisition. For our purpose, the Closed World Asumption is more accurate and more comprehensible to the user. Furthermore, the whole system is a database-based solution, while OWL ontologies are file based. However, there are ways to store OWL ontologies in relation databases [19], but this solution includes transformations from OWL files to SQL statements. This would require to user to define his domain of application in an OWL editor, which is considered tricky, even for experienced users, and then, transfer the whole ontology into the database. So, this approach is not applicable for our purpose, either.

For a more detailed argumentation on OWL ontologies and the described system the reader is referred to paper Girardi et al. [17].

### B. System Architecture

The system is written in object-oriented PHP [20] and uses a MySQL [21] database for data storage. The meta model entities are implemented as PHP classes that allow the creation of structures (entity, attribute, relation, type) and data (record,

 ISBN: 978-1-61208-283-7 

reference, value, user, usergroup). A central main class controls the collaboration of the classes with the user interface. The standard user interface is controlled by the PHP class *GUI*, which encapsulates most of the code for the generation of the HTML output. Furthermore, numerous system functions such as record searching, record tree, calendar, charts, data import, reporting, etc., are provided.

Basically, the system architecture consists of three main parts

*1) Meta Model:* Every entity of the meta model is represented as a separate class in our system.

- **User:** A user represents a physical user that accesses the data and meta data of the system. A user must have a unique name and password with which he is authenticated to the system. Every data manipulation is tracked based on the user that has performed the action. Every user is member of a user group that can define special access permissions. A user owns the records he as created.
- **Entity:** An entity represents a class of objects with equal properties. It has a unique name and it has multiple attributes of different types and names. In contrast to the common relational database paradigm we have implemented the object-oriented inheritance paradigm, thus entities can inherit attributes from other entities. Moreover, we distinguish between tree entities and lookup entities. Tree entities are affected by the end user data acquisition, whereas lookup entities are used for selection boxes of attributes in tree entities. Each entity defines an access level that can either be User, Group or Everyone. Dependent on this access level different users see or do not see records of this entity. If the level is set to User, every user only sees records he has created. If access level is Group, a user has access to all records that have been created by users of his group.
- **Attribute:** An attribute represents one field of data inside an entity. It corresponds to a table field in a relational database but has more properties. An attribute has name and a label. It can also have a group name for grouping multiple attributes in the GUI. Attributes also define their order of appearance in record captions, record tables and selection dropdown lists. Attributes can also have different colors on the GUI and an attribute defines the valid range for numerical values. A regular expression for data validation can also be defined. Attributes can have a tooltip that is display in the GUI and they can have a link with further information that will also be displayed as info button in the GUI (see Figure 3). Relations are treated as attributes of type *Foreign Key*.
- **Type:** The system provides several different types for attributes of entities. The common types are ID, integer, float, string, text, date, time, file. For consistency reasons, we do not allow user-defined types. An attribute as selection of a pre-defined set of choices can be created by adding a relation to a lookup entity.
- **Relation:** A relation represents a 1:n connection between entities just as in a relational database schema. Our relation can also represent an m:n connection resulting in a multi-select list element in the GUI (see attribute 'Others' in Figure 3).
- **Crosslink:** A crosslink is a weak reference between records that are not directly part of a relation. Thus, our system allows to create links between records on different levels of the relational hierarchy. This helps linking information across the hierarchy.

*2) Auxiliary System Functions:* The system provides extra data analysis and visualization features that are integrated in the user interface.

- **Search:** The class RecordSearch provides vast search functionality for records in the system. It is also capable of displaying a user interface for defining and storing search queries.
- **Charts:** Every list of records can be used to fill a 2D chart that can then be displayed anywhere in the user interface. There are several chart types available like line chart, bar chart, pie chart, self organizing maps (SOM), etc.
- **Reports:** Often used queries can be stored as reports for quick reuse in multiple places in the user interface. Reports can be generated quickly and present their results in table format where every cell can be a customized query field.
- **Dependency:** Our system allows to define dependencies between records, which can either result in hiding selection choices that are not plausible in certain conditions or even in hiding whole attributes under certain conditions. Dependecies can be defined over multiple levels in the entity hierarchy providing a powerful tool for increasing data acquisition quality. Dependencies can also be used to check plausibility of entered or imported data, which allows easy determination of data quality for studies.
- **Import/Export:** Our system provides an I/O interface that currently supports XML import and export, PDF export, CSV import and export.

*3) Web Interface:* The class GUI encapsulates most of the code for generating the HTML user interface of our system. There are functions for generating a header, a footer, a Windows Explorer-like record tree with full navigation functionality, dropdown areas, etc. All data dependent code is generated by the meta model classes themselves. A record can display itself either in a table row or as an HTML form. The representation can be customized by parameters. An entity can display all its records as a table. Clicking on a record row automatically displays the record's HTML form without further implementation.

The system also provides a comprehensive administrator interface for meta model manipulations. Users with administrator access can add/delete entities, manipulate attributes and relations, or manage users.

 ISBN: 978-1-61208-283-7 

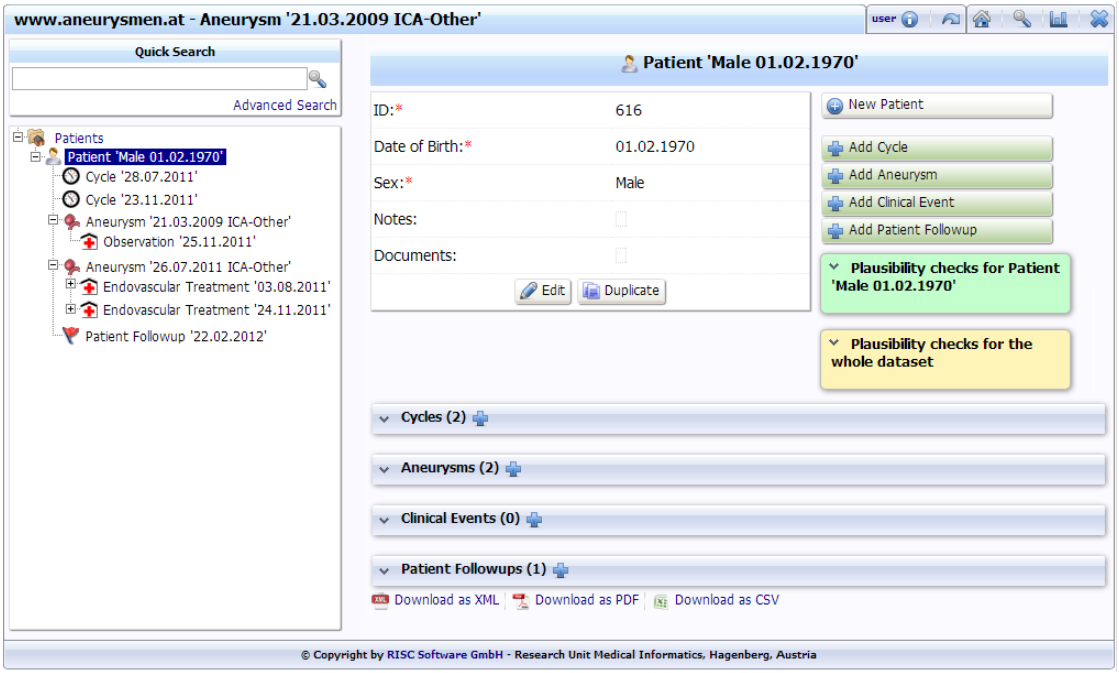


Fig. 2. Standard user interface in cerebral aneurysm registry for a record of entity *Patient* with *Add* buttons for all relations and showing the results of plausibility checks of this record, as well as the record hierarchy navigation tree on left side

The appearance of the user interface (as depicted in Figure 2) can also be customized by the concept of multiple startpages. Every instance of our system can define its own startpage providing almost infinite possibilities. Every instance has a unique configuration file, which defines the database connection and the startpage of the system. Thus, multiple instances of our system can be hosted on one server.

### C. Dynamic GUI Generation

Due to the usage of a meta model as basis for the generation of the user interface, no source code changes are necessary, if the data structure of the system is changed. Administrators can arbitrarily add, delete or modify entities, attributes and relations. The user interface changes instantaneously according to the changes. Every record can display itself on the user interface by showing a table of all attributes of its entity, a collection of 'Add' buttons for every relation and automatic plausibility checks based on the defined dependencies (see Figure 2).

Every record automatically creates an HTML interface for editing all its attributes according to their types. An HTML form is created as table with one row for every attribute of the record's entity. Textual and numeric attributes are displayed as text fields (see attributes 'Admission Date', 'Presentation Date' and 'NIH at Presentation' in Figure 2, whereas relations create a selection box filled with the records of the referenced entity (see attributes 'mRS before Presentation', 'Cycle Presentation', and 'Hunt & Hess' in Figure 2). Our meta model also allows relations of one record with multiple other records, which is displayed as a box of checkboxes with the referenced records in the HTML form of the record (see attribute 'Others' in Figure 2).

Attributes can contain various additional properties according to our meta model such as minumum value, maximum value, regular expression, labels, font colors, tooltip text, information link, ordering, and various others. If entered values mismatch the properties minimum value, maximum value or regular expression, a warning hint is displayed in the HTML user interface.

Below, the records attribute fields and *Add* buttons all existing related records are shown as tables in dropdown areas entitled by the related entity name and the number of related records. Thus, it is possible to quickly navigate through the whole data tree of a root object, which is in our case the *Patient* record. The plus button next to every related entity caption allows the quick creation of new related records of that particular entity. For example, by clicking *plus* next to *Cycles (2)*, the system shows the interface for creating a new record of entity *Cycle* that has automatically set the prior patient as parent record.

## IV. RESULTS - A CEREBRAL ANEURYSM REGISTRY

### A. Definition of a Cerebral Aneurysm

A cerebral aneurysm is the dilation, bulging, or ballooning-out of part of the wall of an artery in the brain. Cerebral aneurysms can occur at any age, although they are more probable in adults than in children and are slightly more common in women than in men. The signs and symptoms of an unruptured cerebral aneurysm will partly depend on its size and rate of growth. For example, a small, unchanging aneurysm will generally produce no symptoms, whereas a larger aneurysm that is steadily growing may produce symptoms such as loss of feeling in the face or problems with the eyes. Immediately after an aneurysm ruptures, an individual may experience such

 ISBN: 978-1-61208-283-7 

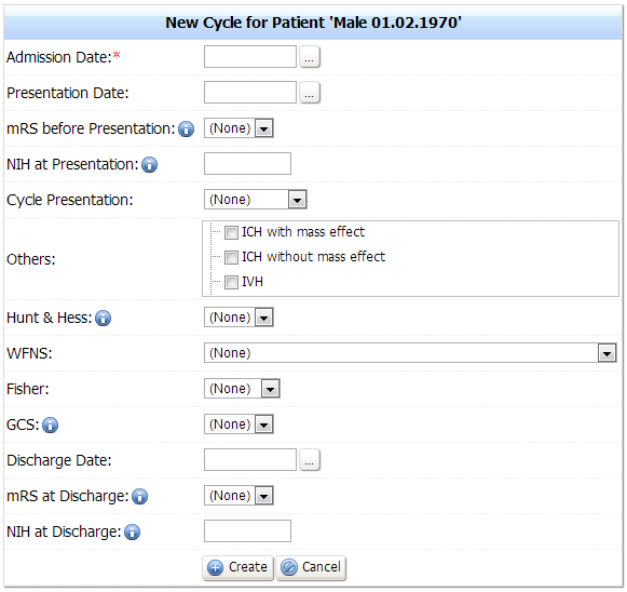


Fig. 3. User interface for new Cycle record after clicking plus on related records caption

symptoms as a sudden and unusually severe headache, nausea, vision impairment, vomiting, and loss of consciousness [22].

### B. Epidemiological Aspects

Intracranial aneurysms occur in 1 to 5 percent in adult population, which results in about 12 million patients in the US. Most of these aneurysms (50 to 80 percent) are rather small and do not rupture during a patient's life time. The estimated incidence of subarachnoidal hemorrhage (SAH) from a ruptured intracranial aneurysm is 1 case per 10,000 persons (in the US). SAH is more common in women than in men (2:1) and the peak incidence is in persons 55 to 60 years old. Although the causes of intracranial aneurysms are not yet know, smoking and hypertension are supposed to have big influence on the development of aneurysms [23].

### C. Purpose of the Registry

The aneurysm registry [24] was established in 2008 by the Institute for Radiology, Landesnervenklinik Wagner Jauregg in Linz, Austria. Its aim is to properly gather and store all relevant information of a patient with cerebral aneurysms. As opposed to hospital information systems, the aneurysm database contains solely medically relevant data in a structured way (no free text or semi-structured information). This allows further automatic processing and analysis of the data, which is used for medical research and internal quality benchmarking.

### D. Data Structure

The data structure is organized in a single rooted acyclic entity graph (figure 4). The top-most entity, in case of the aneurysm data set, is the entity *Patient*. It contains the basic demographic key data (sex, age) and it has four relations to sub-entities. The first kind of sub-entity is *Cycle*. It encapsulates all relevant information about a single hospital stay of the Patient such as admission date and discharge date, and several medical scores at admission and discharge. In the sub-entity *Clinical Event* all complications are recorded. Each complication is linked to a hospital stay by a crosslink relation. The next sub-entity is *Aneurysm*. It contains all information about the aneurysm's morphology and pathology. This entity is the container for all *Treatment* entities where all treatments (observations, surgical or endovascular treatments) are recorded. Each treatment is linked to a cycle by a crosslink relation as well. The last sub-entity of Patient is the *Follow up*, which represents follow-up results over an interval of several years.

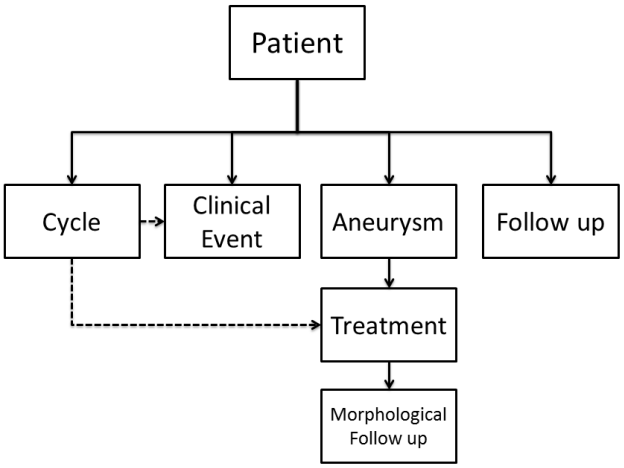


Fig. 4. Data structure of the aneurysm registry (boxes depict Entities, solid arrows depict Relations, dotted arrows depict Crosslinks)

### E. Results

The aneurysm registry was started in 2009 together with Dr. Johannes Trenkler of the Landesnervenklinik Wagner-Jauregg Linz. Since then, over 570 cerebral aneurysms have been recorded, including detailed data on treatments, clinical events, anamnesis, and clinical findings. Based on this data set, several medical dissertations have already been written.

## V. CONCLUSION AND FUTURE WORK

### A. Conclusion

We have shown that our system is capable of storing arbitrary data structures and corresponding data. Once installed, the back-end web interface allows complete customization of the data structure with more possibilities than common relational database systems. The key benefit of our system is the automatically generated front-end based on the data structure stored in the meta model. Therefore, changes to data structure do not affect the underlying source code of the system and can be done after the data acquisition has already started. Acquired data is transformed automatically regarding to the structural changes. Another benefit of our system is the automatic check for plausibility for both entered and imported data. A simple but powerful search interface allows immediate data exploration for users, who can then export their search results as XML, CSV or PDF.

The cerebral aneurysm registry was the first application of our system. Meanwhile, we have successfully instantiated

 ISBN: 978-1-61208-283-7 

the system as a neuro-surgical complication registry, a measurement acquisition for child proportions, a tracking system for eye surgeries, and a benchmarking system for clinical performance. Experience from these applications show that the meta-model based approach allows very low lead times for new data acquisition projects. Given that the data structure is already known - e.g. from a former data collection via excel - the system can be set up and online withing hours. Furthermore, in the first phases of a new project, where, as experience teaches, a lot of little changes are to be made, the ability for the user to perform this changes on their own was a valuable feature. Long-term experience with large data acquisition systems showed that the additional layer of abstraction does not cause any performance bottle necks, although the whole transactional data is actually stored in only three tables (record, value, reference).

### B. Further Research

The system is still subject of further development and research. Currently, we are working on a generic expression engine which allows the user to define arbitrary expressions consisting of existing attributes and operators. These expressions can then be used for complex search queries or for defining logical rules for checking the semantic plausibility. For example, if the size and the weight of a patient are part of the domain specific data structure, then an expression can be created that calculates the body mass index (BMI) for this patient. The BMI can then be used to find patients with a BMI in a certain range or to check if the BMI is in a plausible interval. Furthermore, the BMI can also be used as a new variable for subsequent data analysis, which is the second big subject of research.

We are about to extend the data acquisition system in a way that allows the user to explore his data. The stored meta data helps to decrease the effort the user has to put into data preparation and analysis. Furthermore, the meta data allows to automatize many pre-processing steps for data mining and enables the automatic transformation in external file formats, like the ARFF data format [25] of the Weka machine learning framework. So, like the the meta model based approach reduces the users effort for creating a data acquisition infra-structure it is going to actively support the user with data analysis and mining. For more detailed explanation the reader is referred to [26] and [27].

## VI. ACKNOWLEDGEMENTS

We would like to thank the Landesnervenklinik Wagner-Jauregg of the GESPAG and the federal state of Upper Austria for their financial support. Special thanks go to Dr. Johannes Trenkler and Dr. Raimund Kleiser for their effort in creating and deploying the aneurysm registry presented in this paper.

## REFERENCES

[1] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, *Data Mining: A Knowledge Discovery Approach*. Springer, 1 ed., Feb. 2007.

[2] J. He, "Advances in data mining: History and future," *Intelligent Information Technology Applications, 2007 Workshop on*, vol. 1, pp. 634–636, 2009.

[3] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[4] T. Runkler, *Data-Mining : Methoden und Algorithmen intelligenter Datenanalyse ; mit 7 Tabellen*. Wiesbaden: Vieweg + Teubner, 2010.

[5] M. Bursa, L. Lhotska, V. Chudacek, J. Spilka, P. Janku, and M. Huser, "Practical problems and solutions in hospital information system data mining," in *Information Technology in Bio- and Medical Informatics* (C. Bhm, S. Khuri, L. Lhotsk, and M. Renda, eds.), vol. 7451 of *Lecture Notes in Computer Science*, pp. 31–39, Vienna: Springer Berlin Heidelberg, 2012.

[6] F. Leiner, W. Gaus, R. Haux, and P. Knaup-Gregori, *Medical Data Management - A Practical Guide*. Springer, 2003.

[7] S. Tsumoto and S. Hirano, "Data mining in hospital information system for hospital management," in *Complex Medical Engineering, 2009. CME. ICME International Conference on*, (Tempe, AZ), pp. 1 –5, april 2009.

[8] S. Tsumoto, S. Hirano, and Y. Tsumoto, "Information reuse in hospital information systems: A data mining approach," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pp. 172 –176, aug. 2011.

[9] Hibernate, "http://www.hibernate.org/," May 2013.

[10] Ruby, "http://www.ruby-lang.org/en/," May 2013.

[11] R. on Rails, "http://rubyonrails.org/," May 2013.

[12] J. D. Franklin, A. Guidry, and J. F. Brinkley, "A partnership approach for electronic data capture in small-scale clinical trials," *Journal of Biomedical Informatics*, vol. 44, Supplement 1, no. 0, pp. S103 – S108, 2011. AMIA Joint Summits on Translational Science 2011.

[13] P. A. Harris, R. Taylor, R. Thielke, J. Payne, N. Gonzalez, and J. G. Conde, "Research electronic data capture (redcap)a metadata-driven methodology and workflow process for providing translational research informatics support," *Journal of Biomedical Informatics*, vol. 42, no. 2, pp. 377 – 381, 2009.

[14] T. Zavaliy and I. Nikolski, "Ontology-based information system for collecting electronic medical records data," in *Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), 2010 International Conference on*, p. 125, feb. 2010.

[15] N. Kodama and T. Matsuzaka, "Web-based data acquisition system of wind conditions and its application to power output variation analysis for wind turbine generation," in *SICE-ICASE, 2006. International Joint Conference*, pp. 3747 –3750, oct. 2006.

[16] P. P. S. Chen, "The entity relationship model - toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, pp. 9 – 36, March 1976.

[17] D. Girardi, K. Arthofer, and M. Giretzlehner, "An ontology-based data acquisition infrastructure," in *Proceedings of 4th International Conference on Knowledge Engineering and Ontology Development*, (Barcelona), pp. 155–160, October 2012.

[18] B. Chandrasekaran, J. Josephson, and V. Benjamins, "What are ontologies, and why do we need them?," *Intelligent Systems and their Applications, IEEE*, vol. 14, no. 1, pp. 20–26, 1999.

[19] N. K. Irina Astrova and A. Kalja, "Storing owl ontologies in sql relational databases," *Engineering and Technology*, vol. 29, 2007.

[20] PHP, "http://php.net/," May 2013.

[21] MySQL, "http://www.mysql.com/," May 2013.

[22] NIH, "Cerebral aneurysm information page," April 2010.

[23] J. Brisman, K. K. Song, and D. Newwell, "Cerebral aneurysms," *The New England Journal of Medicine*, vol. 355, pp. 929–939, 2006.

[24] OEGNR, "www.aneurysmen.at," May 2013.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, 2009.

[26] D. Girardi, K. Arthofer, and Giretzlehner, "Ontology-guided data acquistion and analysis," in *Proceedings of 1st International Conference on Data Analytics DATA ANALYTICS*, (Barcelona), 2012.

[27] D. Girardi, M. Giretzlehner, and J. Küng, "Using generic meta-data-models for clustering medical data," in *ITBAM*, (Vienna), pp. 40–53, 2012.

 ISBN: 978-1-61208-283-7 