2015 International Conference on Computational Science and Computational Intelligence

Flexible yet Efficient Management of Electronic Health Records[1]

Bernhard G. Humm
Hochschule Darmstadt – University of Applied Sciences
Darmstadt, Germany
bernhard.humm@h-da.de

Paul Walsh
NSilico Lifescience Ltd.
Bishopstown, Co. Cork, Ireland
paul.walsh@nsilico.com

***Abstract*—This paper presents an approach for managing electronic health record (EHR) data in a flexible yet efficient way. The data model is based on the HL7 Reference Information Model. If clinical requirements change, the data model can be extended by editing a schema spreadsheet. A code generator and the use of state-of-the-art ORM tools allow consistency between EHR application and database. Due to the genericity of the approach, cross-cutting concerns like concurrency control and auditing can be implemented widely.**
The approach has been implemented in form of an EHR framework. As a proof of concept, a subset of a commercial melanoma care EHR application has successfully been implemented.

Keywords—EHR; HL7; code generation; efficiency; flexibility

I. Introduction

Medical information technology has recently advanced in many countries, and enormous amounts of clinical data are already stored as *electronic health records (EHRs)* [1]. At the same time, the domain of clinical information has evolved and expanded rapidly and continues to do so [2]. For example U.S. healthcare data in 2011 is reported to have reached, 150 exabytes, while a Californian health network it is believed to have up to 44 petabytes of data from EHRs, including images and annotations [3],[4]. This leads to a dilemma. On the one hand side, efficient management (storage and retrieval) is necessary to deal with large amounts of EHR data. On the other hand, flexibility is needed to cope with the rapid evolution and expansion of EHR data schemas.

While static approaches provide high efficiency, they lack flexibility. The advantages and disadvantages of dynamic approaches are complementary. In this paper, we present a hybrid approach that combines the advantages of static and dynamic approaches.

Regulations in various countries focus on the privacy / security aspects of EHRs in order to prevent potential abuses (e.g., [5]). However, those aspects are out of this paper's scope.

The remainder of this paper is structured as follows. Section II details requirements for managing EHRs. Section III reviews existing approaches. Sections IV and V detail our solution and describe an implementation. Section VI evaluates our approach. Section VII concludes this paper.

II. Requirements

A solution for managing EHR we expect to meet the following requirements.

1. **General**: The data model shall be capable of handling all kinds of EHR data that may occur in daily clinical practice, e.g. melanoma issues, breast cancer cases, etc.
2. **Flexible**: New entities and attributes may be added easily over time without the necessity of complex data migrations during production.
3. **Efficient**: Storage and retrieval of large amounts of EHRs is efficient in terms of access performance (read / write) as well as storage space.
4. **Convenient**: Programming abstractions for creating and querying EHR data (object / relational mapping) and state-of-the-art development support like intelligent code completion shall be supported.
5. **Interoperable**: importing and exporting EHRs from and to other medical applications shall be supported.
6. **Cross-cutting concerns**: Cross-cutting concerns like auditing, traceability etc. shall be supported

III. Related Work

A. Static Approaches

In traditional business information systems development, a static, bespoke data model is developed and implemented for a set of required use cases [6]. The development of static data models for EHR applications is common practice, too. For example, the data model of the open source EHR management

[1] This work was funded by the European Commission, Horizon 2020 Marie Skłodowska-Curie Research and Innovation Staff Exchange, under grant no 644186.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.84

application Caisis ([7], [8], www.caisis.org) contains more than 200 tables with more than 5,000 attributes (columns) in total. Tables are suited to most specific medical use cases, e.g. `DxImageEndorectalUltrasound`. The advantage of the approach is obvious: via database indexing, the performance of the EHR application can be optimized individually for each use case – thus, it is efficient. However, the static nature of the data model induces a high cost when it comes to new or modified use cases. Each new medical procedure in combination requires the extension of the data model and potentially the need to migrate data within running EHR applications.

B. *Entity Attribute Value Approach*

Because medical procedures advance constantly, researchers and practitioners have long tried to alleviate the disadvantages of static approaches by dynamic approaches. The entity attribute value (EAV) approach [9] has been applied in various life sciences applications, e.g., [2], [11], [12], and [13].

In the EAV approach, data are conceptually stored in a single table with three columns: an Entity (the object being described), an Attribute (an aspect of the object being described), and the Value for that attribute. The advantage lies in the flexibility to add entities and attributes at runtime. The drawback of an EAV system is that the data's physical organization is significantly different from the way users conceptualize it (as one column per attribute) [9]. Therefore, accessing EAV data (adding, querying) is inconvenient from a programmer's point of view. Also, access performance may be poor since numerous join operations are required when retrieving whole sets of patient data.

One can say that the EAV approach's advantages and disadvantages are complementary to those of the static approach.

C. *OpenEHR*

openEHR is an open standard that specifies the management and storage, retrieval and exchange of health data in EHRs, with the aim of providing a powerful means of expressing health information so it can be understood and processed wherever there is a need, independently of a reference model.
This is achieved through the use of an "archetype", which provides a place to formally define data definitions. Archetypes are collected into libraries, which are re-usable domain content definitions, which are created, reviewed and published by domain experts. Templates are used in openEHR used to logically represent patient specific data by referencing the correct data definitions from archetypes. This approach enforces standardization of medically defined terms over all layers that allows the reuse of semantically defined terms that are more universally understood. However, openEHR has limited adoption due to the modeling learning curve by clinicians. Ultimately, openEHR can be viewed as an EAV approach at the data level.

D. *HL7 Reference Information Model*

Health Level Seven (HL7, [10], www.hl7.org) is a set of standards for primarily concerned with transferring EHR data between software applications of healthcare providers. HL7 defines a Reference Information Model (RIM), an ANSI approved standard which is described as "the cornerstone of the HL7 development process". However, HL7 does not specify a concrete EHR data model or EHR application.

IV. Flexible yet Efficient Management of EHRs

In this section, we present a hybrid solution for managing EHRs which combines the advantages of static and dynamic approaches and thus is flexible yet efficient and addresses some of the limitations of related work.

A. *The Basis: HL7 Reference Information Model*

Our approach is based on the HL7 RIM. Fig. 1 shows the main base entity classes of the RIM as UML class diagram.

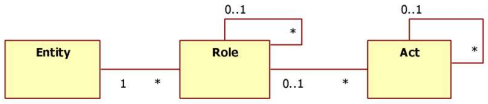

Fig. 1: Main base entity classes of the HL7 RIM

The core of the HL7 RIM is simple and generic in order to support all aspects of EHR. The main base entity classes are:

- **`Entity`** may include persons as well as organizations.
- **`Role`** allows specifying roles which persons and organizations may have in a clinical setting, e.g. patient or consultant. `Role` is separated from `Entity` to allow modelling that one person may have various roles.
- **`Act`** comprises clinical documents, encounters, observations, procedures, etc., i.e., the main EHR data.

For a concrete EHR data model, concrete classes may be defined as subclasses of the base entity classes. See Fig. 2

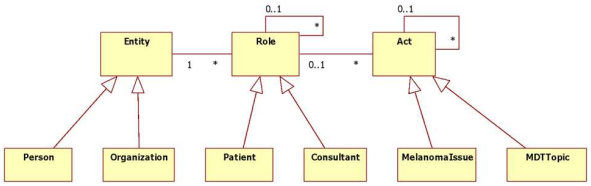

Fig. 2: Classes for melanoma care EHR application

For example, we will use a real-world clinical case study in the field of melanoma treatment. Melanoma are a type of cancer that develops from the pigment-containing cells known as melanocytes and patients can present with multiple classes of melanoma lesions in a wide range of sites on the skin and other organs. The flexibility of the model is utilized by defining the classes `MelanomaIssue` (detail data for a melanoma lesion) and `MDTTopic` (topic to be discussed at a multidisciplinary team meeting) are defined as subclasses of `Act`.

Using this data model, concrete patient data may be represented. See an example as UML object diagram in Fig. 3.

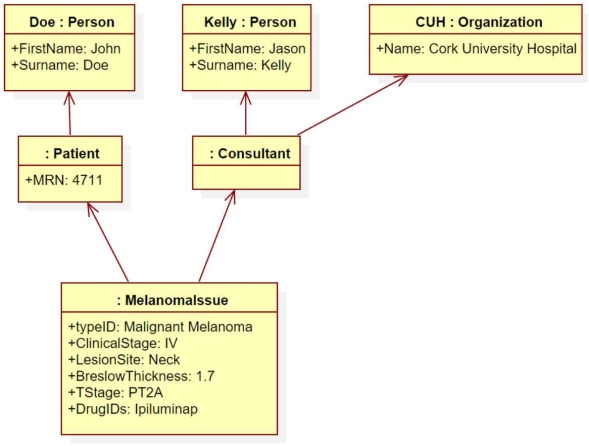

Fig. 3: Sample patient EHR data

Patient John Doe with MRN 4711 has a malignant melanoma on his neck with TStage PT2A. The treating consultant is Jason Kelly of Cork University Hospital. This illustrates that this relatively simple model can be used to capture the essence of patient case. This approach can then easily be extended to a wide range of ad hoc clinical scenarios by the use of straight forward modelling via a spreadsheet tool, as shown in Section B below. Systems such as openEHR on the other hand require third party modelling tools and have a steeper learning curve.

B. *Data Model Configuration*

We support convenient modelling of concrete EHR classes and their attributes in a schema spreadsheet. See Fig. 4.

	A	B	C	D	E
1	Classes	AttributeName	Datatype	Min	Max
50	Person	DateOfBirth	DateTime	0	1
51	Person	FamilyName	String	1	1
52	Person	FirstName	String	1	1
53	Person	Gender	Gender	0	1
54	Consultant	Hospital	Organization	0	1
55	Patient	CNSIssue	CNSIssue	0	1
56	Patient	Mobility	Mobility	0	1
57	Patient	MRN	String	1	1
58	Patient	ResidentialCare	ResidentialCare	0	1
59	Patient	Anticoagulents	Medication	0	*
60	Patient	ImmuneSuppressionDrugs	Medication	0	*
61	Patient	ImmuneSuppressionDrugsDetails	String	0	1
62	Patient	NonAnticogulants	Medication	0	*
63	Patient	NonAnticoagulantsDetails	String	0	1
64	Patient, Consultant	Person	Person	1	1
65	MDTDiscussion	MDTMeeting	MDTMeeting	1	1
66	MDTDiscussion	MDTDiscussionPoint	MDTDiscussionPoint	1	1

Fig. 4: Schema spreadsheet for editing the EHR data model

The schema spreadsheet contains columns for editing classes and their attributes including attribute name, datatype, and cardinality min/max. For example, the class `Patient` contains an attribute `Anticoagulents` of datatype `Medication` with cardinality 0..* (zero to many).

C. *Code Generation*

A code generator transforms the data model specified in the schema spreadsheet into an object-oriented programming language of choice. For example, if the EHR application is implemented in C#, the following source code for class Patient can be generated:

```
public partial class Patient : Role
{
    public virtual CNSIssue CNSIssue { get; set; }
    public virtual Mobility Mobility { get; set; }
    public virtual String MRN { get; set; }
    public virtual ResidentialCare ResidentialCare { get; set; }
    public virtual ICollection<Medication> Anticoagulents { get; set; }
    public virtual ICollection<Medication>ImmuneSuppressionDrugs {get; set;}
    public virtual String ImmuneSuppressionDrugsDetails { get; set; }
    public virtual ICollection<Medication> NonAnticoagulents { get; set; }
    public virtual String NonAnticoagulentsDetails { get; set; }
    public virtual Person Person { get; set; }
}
```

D. *Database Schema*

Using widely used object/relational mapping (ORM) tools like, e.g., .NET Entity Framework, the generated classes can be mapped to tables of a relational database. ORM tools provide different strategies for mapping class inheritance hierarchies to tables: (a) table per hierarchy, (b) table per type, and (c) table per concrete class.

Using strategy (a) table per hierarchy, the generated EHR classes are mapped to three database tables according to the HL7 RIM base classes `Entity`, `Role`, and `Act`. See Fig. 5.

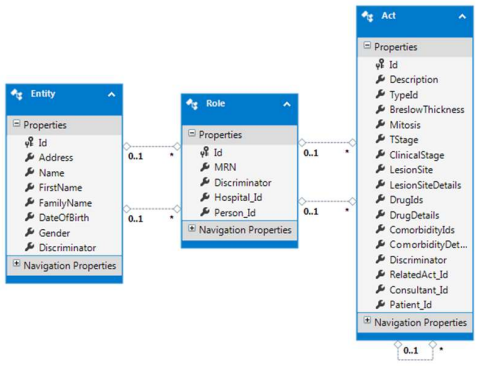

Fig. 5: Database schema

E. *Sparsity*

When using strategy (a) table per hierarchy, the ORM tool generates a discriminator column to discriminate the various subclasses of the inheritance hierarchy. The table contains all attributes of all subclasses.

One effect of this strategy is *sparse tables* since columns for attributes of different subclasses remain `NULL`. See Fig. 6.

	Id	Address	DateOfBirth	FamilyName	FirstName	Gender	Name	Discriminat...
▶	1	NULL	NULL	NULL	NULL	NULL	Cork Univer...	Organization
	2	NULL	NULL	Kelly	Jason	0	NULL	Person
	3	Ballincollig, ...	21.01.1965 ...	Doe	John	0	NULL	Person

Fig. 6: Sparse database tables

Columns like `FirstName`, `FamilyName`, and `DateOfBirth` are relevant for persons only and remain `NULL` for `Organisations`; and vice versa for Columns like `Name`.

For very large data sets, sparse tables could result in poor disk space efficiency. However, state-of-the-art databases like, e.g., MS SQL Server offer sparse column options that optimize space efficiency on the database level.

F. Extensions

1) m:n Relationships

If m:n relationships between EHR classes are needed, the simple data model shown in Fig 1 is not sufficient. The HL7 RIM, therefore, provides additional relationship classes `Participation`, `RoleLink` and `ActRelationship`. See Fig. 7.

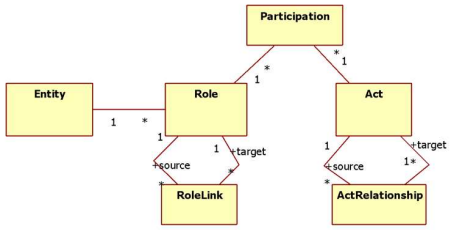

Fig. 7: Relationship classes in HL7 RIM

2) Optimistic Concurrency Control

Since EHR applications are multi-user applications, conflicts may occur if several users, edit the same EHR concurrently. Optimistic concurrency control can be implemented by adding a timestamp attribute to a common superclass of classes `Entity`, `Role`, and `Act`. Due to the generic data model, the optimistic concurrency control can be implemented generically in an EHR application.

3) Traceability

For EHR applications, auditing requirements apply to allow tracing all changes made to an EHR over time. Due to the generic data model, a data historization concept can be implemented generically in an EHR application. See Fig. 8.

Entity

ID	FirstName	Surname	Address
1	John	Doe	Cork
			Dublin

EntityHistory

User	Time	Action	ID	FirstName	Surname	Address
Kelly	2/2/15	create	1	John	Doe	Cork
Kelly	2/9/15	update	1			Dublin

Fig. 8: Historization concept

For the main EHR tables `Entity`, `Role`, and `Act`, historization tables `EntityHistory`, `RoleHistory`, and `ActHistory` are implemented. They contain the same attributes plus, additionally, the attributes `User`, `Time`, and `Action`. When a new row is inserted into the `Entity` table, a new row is added to `EntityHistory` as well. It contains all attribute values of the `Entity` row plus the additional information of which user performed the create action at which time. Whenever the `Entity` row is updated, a new entry is made to the `EntityHistory` table. Again, the action (update) and user and time are recorded. However, this time only the modified attributes (e.g., `Address`) are being stored.

The entity history table allows querying all modifications to individual EHRs.

V. Implementation

The concept as well as the extensions described in the last section have been implemented as an EHR framework. C# was chosen as the programming language, using the .NET Entity Framework as well as MS SQL Server as the database. As a proof of concept, an EHR application for melanoma treatment has been implemented on top of this framework.

See Fig. 9 for a screenshot.

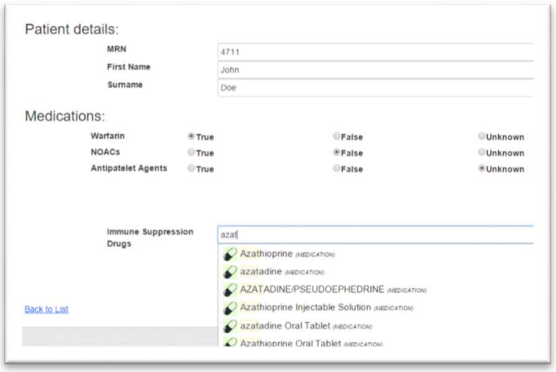

Fig. 9:Melanoma care EHR application

From a programmer's point of view, all EHR data can be accessed as C# objects. For example, creating an EHR record programmatically is as follows.

```
var jasonKelly = new Person() { FirstName = "Jason", FamilyName = "Kelly" };
var cuh = new Organization() { Name = "Cork University Hospital" };
var consultant = new Consultant() { Person = jasonKelly, Hospital = cuh };
var person = new Person() { FirstName = "John", FamilyName = "Doe",
  DateOfBirth = new DateTime(1965, 1, 21) , Gender = Gender.MALE };
var patient = new Patient() { MRN = "4711", Person = person };
EntityDAO.createAll(new List<Entity>() { cuh, jasonKelly, person });
RoleDAO.createAll(new List<Role>() { consultant, patient });
```

Queries can be implemented conveniently using .NET LINQ. Example:

```
public static List<Patient> findPatientsByMRN (String MRN)
{
     var result = Db.Roles.OfType<Patient>().Where(
         p => p.MRN == MRN
         );
     return result.ToList();
}
```

VI. Evaluation

We now evaluate the solution presented with respect to the requirements specified in Section II.

1. **General**: The data model is, indeed, capable of handling all kinds of EHRs that may occur in daily clinical practice. This is due to the use of the HL7 RIM which

itself can be (and has been) applied to clinical data of all kinds.

2. **Flexible**: New entities and attributes may be added easily by editing the schema spreadsheet. The code generation ensures up-to-date classes. The ORM tool ensures up-to-date database tables. Complex data migrations during production can be avoided if attributes are added only.

3. **Efficient**: Retrieval of EHR data via queries is efficient since database columns can be indexed as in traditional, static business information system data models. Multiple join operations as in EAV implementations are avoided. Potential space inefficiencies due to sparse tables can be avoided by sparse column features provided by relational database systems like, e.g., MS SQL Server.

4. **Convenient**: Application programmers can conveniently handle EHR data as objects in the object-oriented programming language of choice. Programming abstractions for creating and querying EHR data (object / relational mapping) and state-of-the-art development support like intelligent code completion is supported.

5. **Interoperable**: Importing and exporting EHRs from and to other medical applications is well supported since our approach is based on HL7.

6. **Cross-cutting concerns**: Cross-cutting concerns like auditing, traceability etc. can be supported generically due to the generic data model. For example, the code for generically identifying the modifications to an EHR record comprises less than 50 lines of C# code.

VII. Conclusions and Future Work

In this paper, we have presented an approach for managing EHRs in a flexible yet efficient way. Advantages of static and dynamic approaches are combined using a hybrid approach. It is based on the HL7 RIM and comprises data model schema editing using a spreadsheet, code generation, the use of state-of-the-art ORM tools and database management tools. Due to the genericity of the approach, cross-cutting concerns like concurrency control and historization can be implemented generically.

The approach has been implemented in form of an EHR framework using MS .NET technology. As a proof of concept, a subset of a commercial melanoma care EHR application has been implemented prototypically. The proof of concept was successful since all data modelling challenges of the commercial application could be handled with the EHR framework. The resulting data model is more compact (less than 70 attributes compared to more than 100 attributes in the commercial application), due to reduced redundancy. The application exhibits good performance.

Therefore, the next step will be to migrate the commercial EHR application to the new EHR framework.

References

[1] Yamamoto K, Sumi E, Yamazaki T, Asai K, Yamori M, Teramukai S, Bessho K, Yokode M, Fukushima M: *A pragmatic method for electronic medical record-based observational studies: developing an electronic medical records retrieval system for clinical research*. BMJ Open 2012, 2:e001622.

[2] Anhøj J: *Generic Design of Web-Based Clinical Databases*. Journal of Medical Internet Research. 2003;5(no. 4):e27.

[3] IHTT: *Transforming Health Care through Big Data Strategies for leveraging big data in the health care industry*. http://ihealthtran.com/wordpress/2013/03/iht%C2%B2-releases-big-data-research-report-download-today/. 2013 (accessed 2015)

[4] Bates DW, Saria S, Ohno-Machado L, Shah A, Escobar G: *Big data in health care: using analytics to identify and manage high-risk and high-cost patients*. Health Affairs 33.7 (2014): 1123-1131.

[5] Blumenthal D, Tavenner M: *The "meaningful use" regulation for electronic health records*. New England Journal of Medicine 363.6 (2010): 501-504.

[6] Simsion G: *Data Modelling – Theory and Practice*, Technics Publications, LLC, New Jersey, 2007.

[7] Fearn P, Sculli F: *The CAISIS Research Data System*. In Biomedical Informatics for Cancer Research. Springer US; 2010:215-225.

[8] Fearn P, Regan K, Sculli F, Fajardo J, Smith B, Alli P: *Lessons Learned from Caisis: An Open Source, Web-Based System for Integrating Clinical Practice and Research*. Computer-Based Medical Systems, IEEE Symposium on Los Alamitos, CA, USA: IEEE Computer Society; 2007, 633-638.

[9] Marenco L, Tosches N, Crasto C, Shepherd G, Miller PL, Nadkarni PM: *Achieving evolvable Web-database bioscience applications using the EAV/CR framework: recent advances*. J Am Med Inform Assoc 2003, 10:444-453.

[10] Beeler GW: *HL7 Version 3—An object-oriented methodology for collaborative standards development*. International Journal of Medical Informatics, Volume 48, Issues 1–3, February 1998, Pages 151–161.

[11] Huff SM, Berthelsen CL, Pryor TA, Dudley AS: *Evaluation of a SQL model of the help patient database*. Proc Symp Comput Appl Med Care. 1991:386–90.

[12] Johnson S, Cimino J, Friedman C, Hripcsak G, Clayton P: *Using metadata to integrate medical knowledge in a clinical information system*. Proc Symp Comput Appl Med Care. 1990:340–4.

[13] Nadkarni PM, Brandt C, Frawley S, et al.: *Managing attribute-value clinical trials data using the ACT/DB client-server database system*. J Am Med Inform Assoc. 1998;5:139–51.