Model Driven Engineering for Large-Scale Clinical Research

Jim Davies, Jeremy Gibbons, David Milward, Seyyed Shah, Monika Solanki, and James Welch
Department of Computer Science, University of Oxford
Email: `Firstname.Lastname@cs.ox.ac.uk`

***Abstract*—Scientific progress in medicine is increasingly dependent upon the large-scale integration of data from a range of sources. This data is acquired in different contexts, held in different systems, and subject to different processes and constraints. The value of the integrated data relies upon common data points having a consistent interpretation in each case; to achieve this at scale requires appropriate informatics support. This paper explains how a model-driven approach to software engineering and data management, in which software artefacts are generated automatically from data models, can be used to achieve this. It introduces a simple data modelling language, consistent with standard object modelling notations, together with a set of tools for model creation, maintenance, and deployment. It reports upon the application of this approach in the provision of informatics support for two large-scale clinical research initiatives.**

I. Introduction

Scientific progress in medicine consists principally in the introduction of new diagnostics and new therapeutics: new ways of figuring out what is wrong, and new ways of doing something about it. The causes of disease, the expression of symptoms, and the response to treatment, can be extremely complex, involving many levels of biology—genomics, proteomics, metabolomics—and a wide variety of lifestyle and environmental factors.

To obtain the evidence to support the introduction of a new diagnostic or therapeutic, while taking account of the variations in biology and environment, we need to make detailed, comparable observations of a suitably large number of individuals. These observations will be made by different people, under different circumstances, and stored and processed in different information systems. A significant degree of coordination is required to achieve an adequate combination of detail and comparability.

One means of achieving this involves prior agreement upon a prescribed dataset: a precise specification of the observations needed to answer a particular question. An information system is then implemented or configured to match this dataset, and staff are trained to acquire data to match the prescribed interpretation. This can be time-consuming and expensive. Furthermore, the data acquired is unlikely to be re-used: not only is the dataset as a whole tailored towards a particular question, but the same may be true of the individual observations.

It should be clear that this is unnecessary. A dataset can be composed by drawing upon, and adding to, a library of existing data definitions. As a precise specification, it can be used as the basis for the automatic generation or configuration of an information system for data capture, reducing costs while enhancing the value of the data obtained. The enhanced value comes in part from the fact that some of the data has been recorded against existing definitions, but also from the fact that definitions are readily available—in a re-usable, computable form—for all of the data collected.

Costs can be further reduced, and scientific progress accelerated, by re-using data already collected: in other scientific studies; in the course of care delivery; or from other sources, including new sensor technologies. Precise data definitions are important here also: we need to know whether the data already collected is fit for purpose, whether its interpretation is consistent with the analysis that we intend to perform. In most cases, these definitions will be created retrospectively, by progressively adding information about the context of collection, and any subsequent processing, until the interpretation of the data is clear.

Having created precise definitions for existing data, we can enhance the value of the data, and reduce the costs of research, by making these definitions available in a standardised, computable form. In designing and proposing a new clinical study, researchers can take account not only of what questions have been asked before, but also of what data already exists, in other research databases or in clinical information systems. In this way, we can eliminate unhelpful variation in study design and unnecessary duplication of effort in data collection.

Furthermore, if we have both precise definitions of existing data, and precise definitions of data requirements for a study, then we can provide automatic support for aspects of study design and delivery: definitions can be compared, queries can be generated for data extraction, and—if summary metadata on the contents of existing databases is available—the feasibility of a particular study can be assessed in advance.

Scientific progress in medicine is increasingly dependent upon this kind of automatic support for data management and integration. Without it, we will be unable to assemble the evidence needed—detailed, comparable data on thousands or millions of individuals—to produce new insights, to validate new discoveries, and to support the introduction of new diagnostics and therapeutics into clinical practice.

This paper explains how a model-driven approach to software and data engineering can provide the support required for large-scale clinical research. It introduces a simple data modelling language, consistent with standard object modelling notations, together with a set of tools for model creation, maintenance, and deployment. It then reports upon the experience of applying these tools within two large-scale clinical research initiatives.

One of these initiatives involves the re-use of data captured in existing clinical information systems. In the area of translational research, in which new innovations are developed and evaluated in the context of clinical practice ('from bench to bedside'), the data needed to support the science is often the same as that needed to support high-quality care delivery and service improvement. In existing clinical systems, however, the same observation may be recorded in many different ways, and there are significant challenges.

The other involves the development of new information systems for data acquisition and management, as well as the customisation and re-use of existing systems wherever possible. Here, the emphasis is upon defining new, generic datasets, in specific therapeutic areas, with the aim of supporting a wide range of studies using the data collected. A particular challenge for informatics development and data re-use stems from the constant updating of these datasets to take account of new knowledge, new constraints, and new objectives.

Together, these initiatives provide an initial validation of the approach. The software tools are in use by clinical researchers, rather than the software engineers responsible for their design. Data is being collected to the definitions that they have produced, using software that they have generated. It is too early to undertake any quantitative, comparative evaluation; however, it is our hope that a report of the approach taken, and the experience gained thus far, will be useful to those working in the field of automated software engineering.

II. Data Models and Metadata

A. Data models

A dataset definition for clinical research will consist of a number of different parts, each of which declares a set of related data items. Typically, this will be a set of data items that would be collected together: in the completion of a 'case report form' in a study, or in the report of a clinical event. These parts may be 'repeating', in that there may be more than one form of that type completed, or more than one event of that type reported upon.

A data item declaration should explain not only the name under which values are to be stored, but also the type of those values. If the type is numeric, then the unit of measurement should be given. If the type is an enumeration, then the intended interpretation of each value should be explained. Finally, the parts of the dataset may be connected or related to one another, and these relationships may have constrained multiplicities.

It should be clear that a dataset definition can be represented as a class diagram or object model. Data items can be introduced as attributes, parts of the model as classes, and relationships between classes as associations—complete with multiplicities. Data types and enumerations can be used to support attribute declarations, and hence data item definitions; constraints can be used to express range restrictions and intended relationships between data items.

Class diagrams can be used also to provide precise definitions for data sources: existing clinical systems or research databases. These systems may be implemented using a combination of technologies but, assuming that the value of the data justifies the effort required, precise object models can be created to describe the interfaces that they might present for data re-use.

B. Models as metadata

Having constructed precise object models to describe datasets and data sources, we can use them as a basis for automated software and data engineering. This involves the use of models as metadata in three respects: for software artefacts that are generated or configured; for data that these artefacts store or process; and for other models, capturing relationships between data points declared in different models, and hence the data stored or processed against them.

For the first of these, we need only maintain the relationship between the implementation and the model that was used to generate or configure it. If the model is stored in a repository or *model catalogue* and published, then we have only to create a link between the implementation and this published instance. This link may be created automatically in the case where the model is generated, partially or completely, from an existing implementation: for example, from the schema of a relational database.

For the second, a link must be created between each data point, or collection of data points, and the corresponding declaration in the data model. The model catalogue then requires some support for the language in which the data model is expressed, sufficient to support the creation and manipulation of links to specific components: in particular, to specific classes and attributes. If the data is acquired or processed using a software implementation for which a data model has already been registered, then that model may provide a basis for the automatic creation of a link to the corresponding declaration.

For the third, we need the same ability to create and manipulate links between components of different models. Instead of links between model and implementation, however, we have links representing different kinds of relationships: provenance and re-use of model components; versioning of models; and, most importantly, *semantic relationships* between attributes or classes—assertions that one attribute or class has the same meaning or interpretation as another. This is precisely the information needed to support automation in data discovery, integration, and re-use.

C. Data Model Language

The core features of our modelling language are familiar: classes, associations, attributes, constraints, enumerations, data types. However, our interpretation of these features is narrower than that set out in, for example, the specifications of the Unified Modeling Language (UML) [1] or Ecore [2]. For this reason, and because of the need to refer explicitly to models and capture semantic relationships between models, classes, and attributes, we define a domain specific language for data models.

We extend this language to include additional modelling features pertaining to certain kinds of implementation artefacts, such as forms, messages, databases, and documentation. This is necessary only when the feature in question may have a bearing upon the data semantics, and may thus serve as a source or target for a link describing a property or relationship.

For example, a section of a form, corresponding to one or more classes of data, might comprise information about allergies. We may wish to link this section to a term in an ontology recording this fact. This makes it easier for someone interested in models of allergy information to locate and possibly re-use the definitions; it also adds context to any semantic links to data classes and attributes associated with that section.

The generated or configured part of an implementation may include features not described by the data model. This will be the case wherever a model-driven or generative approach has been adopted in the development of an application. In this case, the artefact generated from or represented by the data model is itself a model, in a different modelling language. The use of a data model, in our domain specific language, as metadata for this other model should come as no surprise: it is no different from the use of a model as metadata for program source code.

III. Implementation

A. Data Model Language

A UML metamodel for the data model language is shown in Figure **??**.

The key concept behind the language is that of a datamodel, a datamodel is a model composed of dataclasses and dataelements which holds structured data in a particular subject area. The *models catalogue* toolkit is essentially a registry for these *datamodels*, each one being registered with a version number. The datamodel maintains the structure and relationships between a set of dataclasses and dataelements, each of which essentially represent a *data concept*. Dataclasses are in essence very similar to *entities* in entity-relational diagrams, and to *classes* in object-orientated programming; they are simply data-structures. They can be composed from other dataclasses, or from dataelements. Dataelements are atomic entities which can represent a concept or a component of a concept, they can exist in a datamodel independently or as a compositional attribute inside a dataclass.

We have a built a version of the language using the Eclipse Modelling Framework (EMF [**?**]), basing the various entities on Ecore [2] classes. A simplified overview model, without attributes and methods, showing the Ecore model for this DSL is shown in Figure 1.

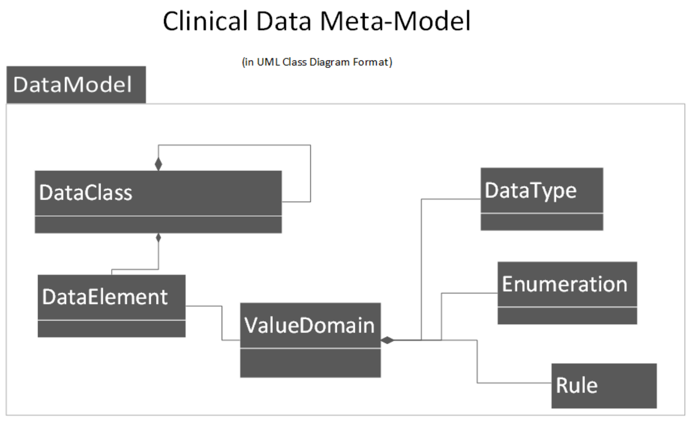


Fig. 1. Overview of Metamodel

1) DataModel: A datamodel is a grouping or containment entity which groups a set of *DataClasses* together. DataModels can be thought of as datasets, or even database schemas, very often in the medical domain they are defined either by XML Schema definition files, or by equivalent schemas written in Excel. DataModels are collections of *ConceptElements* which in turn can be either *Data Elements* or *Classes*. There is no real notion of composition or multiplicity, a instance of a DataModel can contain an instance of a Data Element or not as required by the instance. DataModels are named, have a description and have a version identity.

2) DataClass: A DataClass is a grouping or collection of *attributes* which can be data elements or classes, the attributes are currently *mandatory*, so that DataClass with 5 attributes must have those 5 attributes instantiated in an instance for it to be considered of that DataClass. A DataClass in our meta-modelling language is so named to differentiate it from the term *Class* as used in object oriented programming languages, the main difference being that it captures the structural rather than behavioural aspects of a class. DataClasses represent *Concepts*, and can be *Generalized* into a hierarchy, giving some of the benefits of inheritance to the language. In essence it enables users to take dataclasses from one datamodel, build a sub-class with all the previous data points and then add to it with new dataclasses and dataelement.

3) Data Elements: Data Elements can also represent *Concepts* and are by their nature *atomic*. Each data element is related to a value domain on a one-to-one basis, and the relationship is a two-way relationship.

4) Value Domain: A Value Domain is the domain in which the data element is represented, it can consist of one or more *ValueSpecs*. In addition a valuedomain can have zero or more *Measurement Units*, which are descriptive tags which refer to a measurement unit such as kilometers per hour, imperial pounds, or meters.

5) ValueSpec: ValueSpecs are intermediate entities to describe *how* the data will be represented. There are 3 ValueSpecs, the first is a *datatype*, the second is an *enumeration* of a datatype, and the last is a *rule*. A valuedomain will have at least one ValueSpec and at most 3 ValueSpecs which must be of different kinds. The main reason for creating a language for handling the data structures in this manner is not only so that models can be curated, versioned and maintained easily, but that the datamodels produced can then be transformed automatically for use in other heterogeneous systems. To illustrate this figure 2 shows the LEMMA meta-model within a 4 tier abstraction diagram. The layers within this diagram are based around idea put forward by the OMG([3]), specifically model driven architecture (MDA [4]). The MDA defines n abstraction layers for modelling, although 4-layers are normally used. The M0 layer is *real-world* level, the level at which

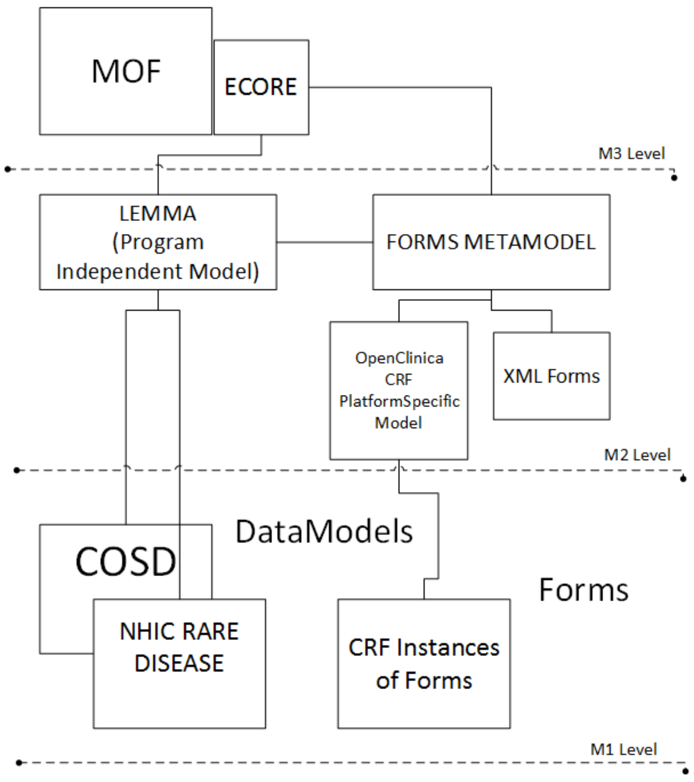
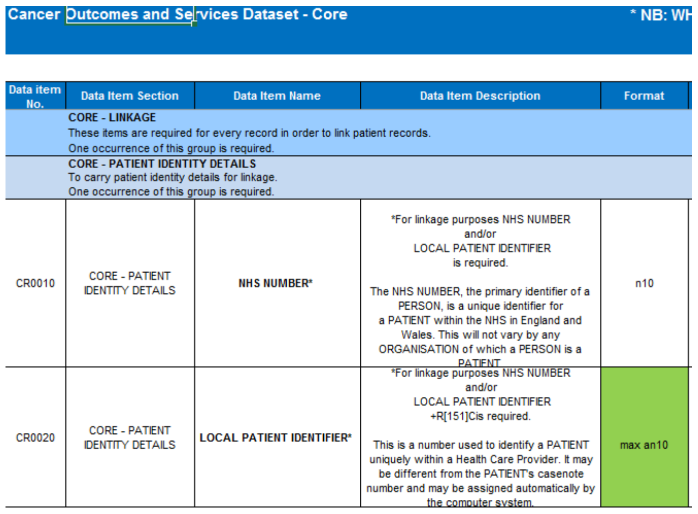


Fig. 2. Datamodel in Excel Format

real world objects exist, people, cars, programs, etc. The M1 layer is level at which programs, such as a java program, or a java class definition exist in their static runtime form. The instantiation of a class occurs at the level below the class, so a java object runs in level M0. In defining datasets or datamodels we are dealing in datamodels which are instantiated at the M1 level, the *Cancer and Outcomes Services Dataset* exists at this level, although their may be many conforming instances running at the M0 level. The datamodel is defined in out meta-modelling language, which exists at the M2 or meta-modelling level, and in turn conforms to the more abstract specification known as ECore.

An instance of a datamodel can be written in this DSL in the same way as java code, and figure 3 shows how this code looks in the eclipse development toolkit. In this diagram the excel representation of the *Cancer and Outcomes Services Dataset* shown in figure 3 has been transformed into the meta-model.

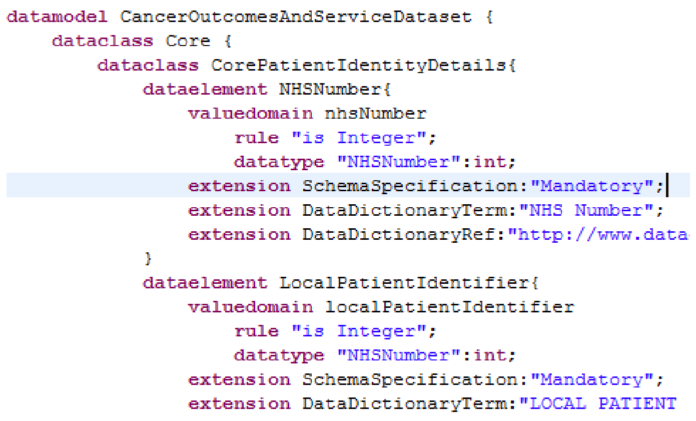


Cancer Outcomes and Services Dataset - Core * NB: WH

Data item No.	Data Item Section	Data Item Name	Data Item Description	Format
	CORE - LINKAGE These items are required for every record in order to link patient records. One occurrence of this group is required.			
	CORE - PATIENT IDENTITY DETAILS To carry patient identity details for linkage. One occurrence of this group is required.			
CR0010	CORE - PATIENT IDENTITY DETAILS	NHS NUMBER*	*For linkage purposes NHS NUMBER and/or LOCAL PATIENT IDENTIFIER is required. The NHS NUMBER, the primary identifier of a PERSON, is a unique identifier for a PATIENT within the NHS in England and Wales. This will not vary by any ORGANISATION of which a PERSON is a PATIENT	n10
CR0020	CORE - PATIENT IDENTITY DETAILS	LOCAL PATIENT IDENTIFIER*	*For linkage purposes NHS NUMBER and/or LOCAL PATIENT IDENTIFIER +R[151]Cis required. This is a number used to identify a PATIENT uniquely within a Health Care Provider. It may be different from the PATIENT's casenote number and may be assigned automatically by the computer system.	max an10

Fig. 3. Datamodel in Excel Format

The DSL is used to represent a model for Cancer data (part of the Cancer Outcomes and Services Dataset [5]) in figure 4, and is being used within the eclipse toolkit. The DSL can be used to transfer datasets between instances of the models catalogue, although other formats such as JSON and XML are also available.



```
datamodel CancerOutcomesAndServiceDataset {
    dataclass Core {
        dataclass CorePatientIdentityDetails{
            dataelement NHSNumber{
                valuedomain nhsNumber
                    rule "is Integer";
                    datatype "NHSNumber":int;
                extension SchemaSpecification:"Mandatory";
                extension DataDictionaryTerm:"NHS Number";
                extension DataDictionaryRef:"http://www.data
            }
            dataelement LocalPatientIdentifier{
                valuedomain localPatientIdentifier
                    rule "is Integer";
                    datatype "NHSNumber":int;
                extension SchemaSpecification:"Mandatory";
                extension DataDictionaryTerm:"LOCAL PATIENT
```

Fig. 4. Datamodel in Eclipse environment

B. Metadata Registration

In designing the language, we gave due consideration to the ISO/IEC 11179 standard for metadata registration. This standard sets out a metamodel for data definitions, together with a set of processes for their registration and publication.

Refer also to the CancerGrid project [6].

The ISO11179 standard uses the notion of a *data element concept*, *a data element*, *a value domain*, and a *conceptual domain*. The standard currently confines itself to the detailed level of concepts and data elements and has no notion of collections of data elements or data element concepts, but instead attaches two attributes: an *object class* and a *property*

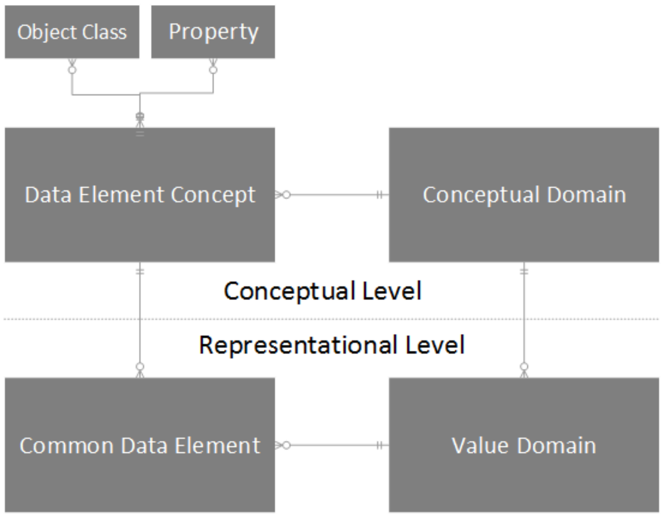


Fig. 5. Core model for ISO11179 Metadata Registry

to each data element concept and these attributes allow the data element concept's to be aggregated or classified. This core model of the ISO11179 is illustrated in figure 5. The data element concept and conceptual domain entities belong to the *conceptual level* whilst the common data element and value domain both belong to the *representational level*.

For example an Integer data-type in a programming language may be used to represent inches in a measurement program, it may also be used to count vehicles in a logistics application. A data element is said to be comprised of a data element concept(DEC) which is its meaning and a value domain(VD) which is its representation.

Entity	ISO Definition	ISO11179 Implementation Guidelines
Data Element Concept(DEC)	An idea that can be represented in the form of a data element, described independently of any particular representation.	A concept that can be represented in the form of a Data Element, described independently of any particular representation.
Common Data Element(CDE)	A unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of attributes.	A unit of data for which the definition, identification, representation and Permissible Values are specified by means of a set of attributes.
Value Domain (VD)	The description of a value meaning.	A description of a Value Meaning.
Conceptual Domain (CD)	A set of valid value meanings, which may be enumerated or expressed via a description.	A set of valid Value Meanings.

The table lists the definitions of the key entities used in the standard Conceptual domains comprise sets of value domains, they provide a collection mechanism of *Value Meanings* which provide representation for a particular data element concepts. Data Element Concepts can then be grouped according to their Object Class, or Property, however whilst this works for a system that is focussed entirely on the metadata units, any working software system will need to group the data elements in a structure that is easily transformed into the components such as Classes and Entities that are used in most information systems.

C. Forms and Automatic Software Generation

By defining the LEMMA metamodel and a similar and dependent Forms metamodel, both at the M" level, and both *Platform Independent* we are able to relate curated data elements to forms elements, and by selecting a group of dataelements and dataclasses automatically generate a set of related forms. This can be done in several ways, the most used way is to generate a platform independent model and then use that model to generate an Excel template.

The Models Catalogue, like many software developments, evolved from a mix of requirements on a number of different projects. Initially a metadata registry was built using an XML database, however problems were encountered with scalability. Most of the language and metamodel developments were carried out using XText and the Eclipse Modelling Framework, resulting in a usable java code base, however usability requirements neccessitated the refactoring of this software using a stack consisting of Grails and Angular JS.

Grails is built on the Spring framework, which is not only proven to be very robust and scalable, but is also relatively easy to implement and so enables quick agile development cycles. Previous implementations using Java/Spring and Java/Roo have proved very time-consuming to experiment with, whereas Grails has proven to be more flexible and easier to experiment with. Domain specific languages (DSLs) can be built on this framework, and this capability offered scope to build a DSL based on the LEM DSL specification and metamodel expanded in the previous section.

The front end user interface was implemented using a combination of HTML with Javascript and CSS, the principal framework used being Angular JS. Communication with the client was carried out using a REST controller, enabling a variety of clients potentially to link up with the Model Catalogue.

GORM was used as a persistence mechanism, with a MySQL relational database as storage, although different GORM adapters made it possible to attach NO SQL datastores such as Neo4J and MongoDB. The full architectural stack is shown in Figure 6

The Grails/GORM framework enabled the Ecore model to generate the basic Grails Domain model, and from that a Groovy DSL was built, using the Builder pattern, to handle transformations internally between different representational languages such as XML and Excel. A series of importers was built for data input from Excel, CSV, and various XML variants. Most XML structures are handled by transforming the XML from its native structure to our internal DSL-based XML structure.

The internal domain model used a basic *Catalogue Element* which was able to link elements via the *relationship* and *relationshipType* classes. The core language model discussed

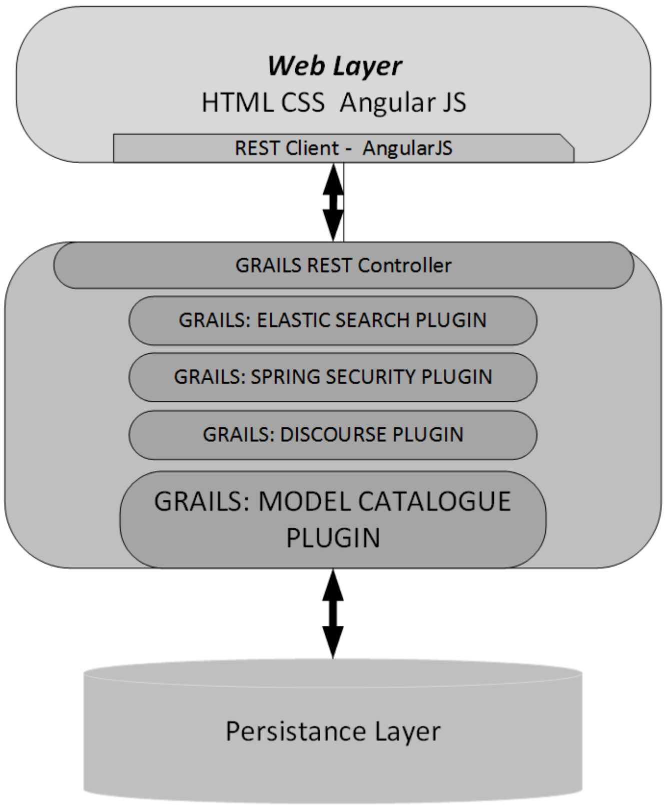


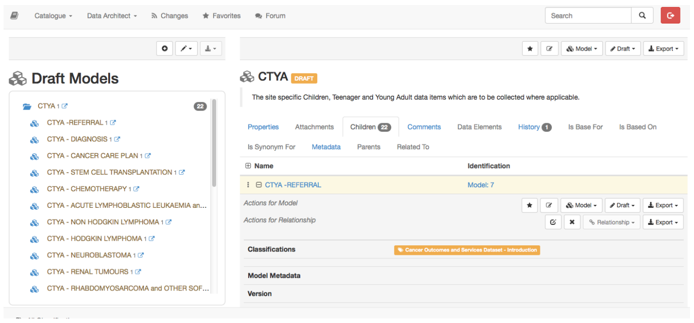
Fig. 6. Overview Architecture

earlier has been enhanced to by allowing user-defined relationships to be added to the core model. Any catalogue element is able to be related to any other catalogue element through a relationship class, this relationship is constrained by the relationshipType object which can prevent different catalogue element types being related, so that a Model cannot directly be related to a say a Datatype Enumeration. Relationship types can be added to the Model dynamically, so that even though the relationship between a Model and Datatype enumeration is prevented initially, a new type could be introduced by an administrator or super user to add in that relationship. The EMF-based tools to automatically generate the whole Models Catalogue code-base using Groovy/Grails/Angular were not available at the start of the project, and although work has been undertaken to build such a toolkit it not yet complete.

The following subsections describe the basic use cases for the Model Catalogue, and how these use cases were implemented.

D. Listing of DataModels

The key use case required in both projects was the ability to catalogue a set of data elements and data classes so that different schema could be compared and curated. Listing is currently carried out using a REST interface which is queried using an Angular client. Figure 7



Draft Models

- CTYA 1 (22)
  - CTYA -REFERRAL 1
  - CTYA - DIAGNOSIS 1
  - CTYA - CANCER CARE PLAN 1
  - CTYA - STEM CELL TRANSPLANTATION 1
  - CTYA - CHEMOTHERAPY 1
  - CTYA - ACUTE LYMPHOBLASTIC LEUKAEMIA an...
  - CTYA - NON HODGKIN LYMPHOMA 1
  - CTYA - HODGKIN LYMPHOMA 1
  - CTYA - NEUROBLASTOMA 1
  - CTYA - RENAL TUMOURS 1
  - CTYA - RHABDOMYOSARCOMA and OTHER SOF...

CTYA DRAFT

The site specific Children, Teenager and Young Adult data items which are to be collected where applicable.

Name	Identification
CTYA -REFERRAL	Model: 7

Actions for Model

Actions for Relationship

Classifications	Cancer Outcomes and Services Dataset - Introduction
Model Metadata	
Version	

Fig. 7. Screen Shot of DataModel Listing

E. forms

Typically the initial data models are entity-centric: for example cancer models are initally built around the entities of *Patient*, *Tumour*, etc. Forms are typically structured around events: *Registration*, *Surgery*, etc. We construct separate models for forms, re-using the elements from the entity-centric model, thus providing a different view upon the same data. (As as aside, the Patient Pathway models interact with these forms models, providing a workflow, ordering form completion).

With respect to forms, we take an approach analagous to that of the Model-Driven Architecture [7]: that of Platform-Specific Models (PSMs) and Platform-Independent models (PIMs). Our platform-independent model in this regard is based on the emerging ISO standard for forms [8]. This provides structure in terms of *sections*, *sub-sections*, *repeating groups*, and so on, but is not specific to any particular implementation. The model is based on previous work on domain-specific languages for forms [9].

Our initial platform-specific target is *OpenClinica*, a tool for supporting data capture for clinical trials. OpenClinica forms are created using a model supplied in Excel Spreadsheet format.

Our generation of XML schema may also be seen as a platform-specific implementation of the forms model. XML schema are used to define message structures, and have a similar structure to the OpenClinica forms, although typically additional data is required in XML schema - context such as the patient for which the data is about, or the person submitting the data; or linking information which may be automatically created during manual input into a web interface.

F. selection of data elements for form generation

For many clinical users one of the key requirement was the ability to generate forms for clinical research which were generated form a single authorative source, and the ability to take data elements, manage them to build a form and then output either a form or an XML representation for use in another system, such as OpenClinica was key to the work carried out.

G. relationships between 2 DataModels

Very often different research groups will arrive at slightly different models for the same or very similar diseased, so another use case for the Models Catalogue was the ability to compare different Data Models, Data Classes and individual Data Elements.

H. Modelling Overview

IV. Experience

A. Re-use of data from clinical information systems

The UK National Institute of Health Research (NIHR) is funding an £11m programme of work across five large university-hospital partnerships: at Oxford, Cambridge, Imperial College London, University College London, and Guy's and St. Thomas'. The aim of the programme is to create the infrastructure needed to support data re-use and translational research across these five institutions.

The programme, the NIHR Health Informatics Collaborative (HIC), was initiated in 2013, with a focus upon five therapeutic areas: acute coronary syndromes, renal transplantation, ovarian cancer, hepatitis, and intensive care. The scope was increased in 2015 to include other cancers—breast, colorectal, lung, and prostate—and other infectious diseases, including tuberculosis.

The key component of the infrastructure consists in repositories of patient data within each of the five institutions. The intention is that these repositories should hold a core set of data for each therapeutic area, populated automatically from clinical systems, together with detailed documentation on the provenance and interpretation of each data point.

Researchers can use the documentation to determine the availability and suitability of data for a particular study. They can use it also to determine comparability across institutions: whether there are any local differences in processes or equipment that would have a bearing upon the combination and re-use of the corresponding data. Once a study is approved, the repositories act as a single source of data, avoiding the need for data flows from individual clinical systems.

The development of the infrastructure required the development of a 'candidate data set' for each therapeutic area, as a core list of data points collected in the course of routine care that would have value also in translational research. Each institution then set out to determine which information systems, within their organisation, could be used to populate each of the candidate data sets: this was termed the 'data exploration exercise'.

The results of the exercise informed further development of the data sets, and data flows were established. To demonstrate and evaluate the new capability, 'exemplar research studies' were initiated in each therapeutic area, using data from all five institutions.

Each institution had a different combination of existing systems, a different approach to data integration, and a different strategy for informatics development. It was not feasible or appropriate to develop a common 'data repository' product for installation. Instead, a set of data models were distributed, and each institution worked to implement these using their own messaging, business intelligence, or data warehousing technologies.

None of the institutions had the capability to provide documentation on the provenance and interpretation of their data in any standard, computable format; the model or metadata aspect of the infrastructure was entirely new. It was this that drove—and continues to drive—the development of a comprehensive model catalogue application.

At the start of the project, teams of clinical researchers and leading scientists were given the responsibility of creating the candidate data sets for each therapeutic area. They did this by exchanging spreadsheets of data definitions in email. This proved to be a slow process, and face to face meetings were needed before any real progress could be made.

It proved difficult to properly represent repeating sections of the dataset—corresponding to investigations or interventions that may happen more than once for the same patient. Researchers resorted to Visio diagrams to try to explain how observations fitted into clinical pathways or workflows—and discovered that there were significant differences between pathways for the same disease at different institutions.

In one therapeutic area, these differences had a profound effect upon the interpretation of certain observations, and the candidate dataset was extended to include additional information on the pathway. Due to the complexity of the pathways involved, this was a time-consuming and error-prone process. Furthermore, the spreadsheets quickly became inconsistent with the Visio diagrams.

The candidate datasets were distributed to the informatics teams at the five institutions in the form of XML schemas. At first, these were created from scratch, rather than being generated. There were many requests for changes to the schemas; these proved difficult to track and coordinate.

The exploration exercise itself was reported by adding columns to the distributed versions of the candidate dataset spreadsheets, listing the information systems containing the data points in question, or suggested alternatives where there were significant differences due to local systems and processes.

This was despite the availability of an initial version of the model catalogue. Researchers and local informatics teams preferred to work with spreadsheets, having little or no knowledge of modelling languages such as UML and no automatic support for model creation and maintenance. It fell to the software engineering team at the coordinating centre to record the datasets and variations in the catalogue.

While it was disappointing to have the researchers still working in spreadsheets, the ability to generate XML schemas from models, and to manage relationships between data items in different models and different versions, proved invaluable. In the second phase of the project, with more therapeutic areas being added, researchers are starting to abandon the spreadsheet mode of working, and are instead maintaining the datasets as data models, in the catalogue.

B. Coordination of clinical data acquisition

The UK Department of Health, through the NIHR and the National Health Service (NHS), is providing funding for the collection for the whole genome sequencing of blood and tissue samples from patients with cancer, rare disorders, and infectious disease. A network of regional Genomic Medicine Centres (GMCs) is being established to collect samples and provide access to genomic medicine across the country.

The results of the whole genome sequencing will

large number of differeto provide access to genomic medicine across the country. Each GMC is a regionpartnership involving all of the hospitals providing care for

Key ideas:

- Data Elements can be used across different DataModels - favourites or shopping cart.
- Unique Identification of DataElements
- Documentation can be automatically generated for review purposes
- Consistency checking carried out on data elements
- Generation of XSDs, XML, and Forms as output, interfacing with OpenClinica.
- Data Provenance

C. Genomics England — reuse of HIC Models

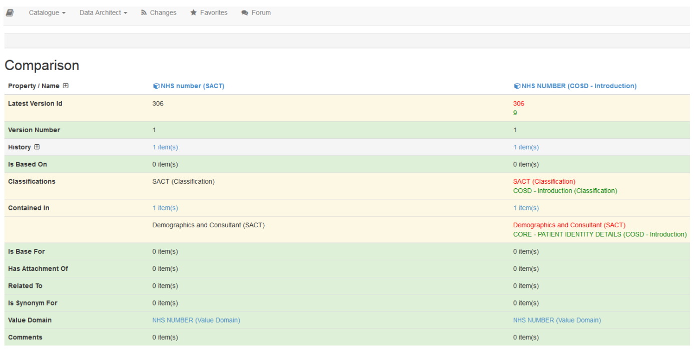
One of the prime tasks of Genomics England (GEL) is to integrate a patient's clinical records with their genome, and use this integrated data to inform new clinical research primarily in the domain of genetic medicine. In order to do this, GEL is assembling core datamodels, initially to classify clinical research in the area of rare diseases and in the area of cancer. Clinicians were asked to assemble datamodels in Excel and XML format, and to attempt to arrive at an agreed datamodel for the two areas. Initially this was carried out by assembling the subject matter experts in a conference room and then going through each item in turn. The process was quite time consuming and difficult: the users were mostly eminent clinicians and extremely busy, so that getting all the contributors in the same room at the same time was itself quite a challenge.

Whilst that process was being carried out, some of these datasets were added to the Model Catalogue, and the subject matter experts were given visibility of the datasets already stored in the Model Catalogue. At this point the Model Catalogue was already loaded with datasets derived from the NHS Data dictionary such as COSD and from the results of the NHIC work. Use of the toolkit resulted in the users being able to very quickly assemble a new dataset from an existing dataset; they were then able to go through each data element in turn and compare it to an existing data element in another datamodel, a process that is illustrated in Figure 8.

For the Genomics England exercise, the Models Catalogue has had a social networking feature added, allowing messaging forums to be automatically generated around a *datamodel* or *dataclass*. This allows users to develop a new draft datamodel, and then

From looking at the datasets we can see that in the first 2 major datamodels produced X% of data elements and dataclasses were derived from existing sources, a fact entirely due to the use of the Model Catalogue in the data curation process. Time to assemble a new datamodel has been cut from months to weeks as a result of having a single shared source for the datamodel, and having the means to electronically comment on and discuss datamodel development in real time as it is happening.

Figure 8



Comparison

Property / Name ⊞	NHS number (SACT)	NHS NUMBER (COSD - Introduction)
Latest Version Id	306	306 9
Version Number	1	1
History ⊞	1 item(s)	1 item(s)
Is Based On	0 item(s)	0 item(s)
Classifications	SACT (Classification)	SACT (Classification) COSD - Introduction (Classification)
Contained In	1 item(s)	1 item(s)
	Demographics and Consultant (SACT)	Demographics and Consultant (SACT) CORE - PATIENT IDENTITY DETAILS (COSD - Introduction)
Is Base For	0 item(s)	0 item(s)
Has Attachment Of	0 item(s)	0 item(s)
Related To	0 item(s)	0 item(s)
Is Synonym For	0 item(s)	0 item(s)
Value Domain	NHS NUMBER (Value Domain)	NHS NUMBER (Value Domain)
Comments	0 item(s)	0 item(s)

Fig. 8. Screen Shot of DataClass Comparison

D. Genomics England - deployment

V. Lessons actually learned?

- metadata registries are not enough - you need models - (we should include examples of how data elements are contextualised - Keith slides) -
- these models should be linked to the implementation, for otherwise it is too expensive to maintain them
- the models should be readable as data manuals - literate modelling - it simply doesn't work to have to click on attributes to understand their meaning and derivation
- this readability applies to the development interface, this is the interface that you want to use (can't do the turn time) - and you need to be able to pair program with domain experts
- you want inheritance within models, and include/import across models (start thinking of the models as code, although this is about managing information in general - managing declarations - it is coincidental that we are generating artefacts from them)
- you may want different types of model for different generated artefacts—depending upon what, exactly, is in the generator code as parameters and additional information—it may be the same set of data points overall, but you might well have a different refactoring in Fowler terms
- version and store the generation code (!)
- tag the generated artefacts with a link back to the model catalogue (XML schemas in HIC!)
- you will want more models than you think: for example, consider the model of a clinical test; now think about how it is dropped into a pathway; you need the pathway model to tell you what the dataset means; it has added further context to the data element definitions (you could flatten this, but that's bad)

- automate aspects of model management—versioning and dealing with multiple models
  - automatically create (and propose) links, including classifications
  - have links for new version of (sequential), refactoring of (parallel), derived from (data concepts across models), same as (strong assertion)
  - use links in model maintenance - note that the definition that this was derived from has changed
  - have publication cycle—a published model is for life
- general lesson: if you want to manage data semantics, you need a compositional approach—none of this central coordination of a single data dictionary, none of this single hierarchy of data elements, none of this element by element description (all the context in the text of the element definition? doesn't work!)
- general lesson: if you are going to manage data at scale, you need data model driven approach
- general lesson: if you have a model driven approach (data model or otherwise) then your management of models has all the same challenges as management of source code (you need an IDE) - really, if you are using models as programs, then you need to support them as programs

VI. Related Work

A range of tools are available for clinical Electronic Data Capture (EDC) including Catalyst Web Tools [10], OpenClinica [11], REDCap [12], LabKey [13] and Caisis [14]. A two-year case-study comparing EDC tools is available in [15], a further study comparing Clinical Trials Management Systems is found in [16]. In the current paper, OpenClinica case report forms (CRFs) forms are generated using the Model Catalogue tool, however, CRFs for any EDC tool may be generated. State-of-the-art in EDC is the research electronic data capture (REDCap) [12] web-based application. The REDCap tool supports meta data capture for research studies providing an online interface for data entry, audit trails, export to common statistical packages and data import from external sources. Like the Model Catalogue, the REDCap system focuses on the clinical metadata. However, REDCap and similar tools differs from the Model Catalogue in several important aspects. Firstly, EDC tools tend to be insular and centralised systems, any data must be imported into and stored within the CEDC database, which acts as a silo. The Model Catalogue aims to provide a platform to support existing data stores and systems within the clinical environment.

There is also a wide-gap in the level of expertise required to operate the tools. The meta-data management (creation, revision, sharing) in CEDC is considered a specialist task [12], [15], requiring experts to initialise a the meta-data per-study. In the Model Catalogue, clinical users have the ability to adapt and modify metadata as needed, and treat models as the central artefact. Effectively, the Model Catalogue follows the same metadata workflow as REDCap, but without the need for modelling experts to develop, adapt or to share the meta data models. REDCap, unlike the Model Catalogue, is not open source so users cannot freely modify the functionality of the system. The Model Catalogue, can be modified to add features and support to maximise the utility of the models. Because the Model Catalogue works at the meta-data level, users can share similar models between organisations and derive new software artefacts from user created metadata models.

Several examples of Model Driven Engineering for e-Heath software are present in the literature. The typical pattern followed in these methods is one formalised in [17], which advocates a multi-phase approach, where data modelling is a separate phase from stakeholder engagement and data integration. This approach is taken in [18], where an Eclipse-based tool is used to develop DSLs to model and generate tools for mobile heath tracking applications. The advantage of the approach is the ability for clinicians to modify the model of the study, using a DSL, and automated creation of applications from the model. A similar approach is taken in the *True Colours* system [19], using the Booster model driven toolkit to derive a patient self-monitoring application for mental health. The Booster approach shows how data tends to be managed better by a model driven process, which leads to higher quality, reusable data. In [20], the authors present experiences on the use of Model Driven Engineering to implement an application for path-based stroke care; amongst the experiences included using existing ontological models, where possible and using an adoptive modelling toolkit. Unlike these systems, in the model catalogue, a meta-data oriented approach is taken so the applications created using models can be made interoperable with existing data, standards and systems. Rather than simply using MDE to develop stand-alone systems, MDE processes are used in the management of clinical trials meta-data from which software is derived.

In the Model Driven Health Tools (MDHT) [21] project, the HL7 Clinical Document Architecture (CDA) standard [22] for managing patient records is implemented using Eclipse UML tools [23]. The benefits of applying model driven engineering are clear, modelling tools are used to model the CDA standards and interoperable implementations of the standard are automatically derived from the models. In principle, this is similar to the approach in the model catalogue, where the CDA meta-data can be represented and implementations derived. However, MDHT supports only the CDA standard, whereas the model catalogue can interoperate with any meta data standard. The CDA standards are large and complex, in [24], a model driven approach is advocated to simplify the HL7 CDA. This is supported by three case studies: the NHS England Interoperability Toolkit (ITK); simplification of US CDA documents and the Common Assessment Framework (CAF) project for health and care providers in England. The paper outlines the benefits of simplifying the CDA standard for practical applications. The model catalogue supports similar simplifications, where large and complex meta data schemes are simplified by mapping only relevant meta data in the generated artefacts.

Several efforts have addressed the representation of ISO11179 as ontological models for enabling data integration across metadata registries (MDRs). In [25] the authors describe a federated semantic metadata registry framework where CDE (Common Data Elements) are exposed as LOD (Linked Open Data) resources. CDEs are described in RDF, can be queried and interlinked with CDEs in other registries using SKOS. An ISO1179 ontology has been defined as part of the framework and the Semantic MDR has been implemented using the Jena framework. In [26] the authors present the Clinical Data Element Ontology (CDEO) for unified indexing and retrieval of elements across MDRs. Concepts in CDEO have been organised and represented using SKOS. In [27] the authors present case studies for representing HL7 Detailed Clinical Models (DCMs) and the ISO11179 model in OWL. A combination of UML diagrams and Excel spreadsheets were used to extract the metamodels for 14 HL7 DCM constructs. A critical limitation of this approach is that the transformation from metamodels to their ontological representation in OWL [1] is based on a manual encoding.

Ontology repositories can be considered closely analogous to model catalogues they provide the infrastructure for storing, interlinking, querying, versioning and visualising ontologies. Relationships capturing the alignments and mappings between ontologies are also captured allowing easy navigability.

Linked Open vocabularies[28] provides a service for discovering vocabularies and ontologies published following the principles of linked data. Besides the above features, it provides documentation about ontologies automatically harvested from their structure and identifies dependencies.

Apache Stanbol[29] provides a set of reusable components for Semantic content management. The Apache Stanbol Ontology Manager provides a controlled environment for managing ontologies and ontology networks. Main components of the manager include (a) Ontonet, a Java API for the construction and management of ontology networks from the ontology knowledge base. (b) Registry, an RDF resource that provides descriptions of ontology libraries in the knowledge base. The registry also provides a management API for administrators to pre-configure sets of ontologies loaded in the ontology libraries.

KAON2[30] provides an API infrastructure for managing OWL-DL, SWRL and F-Logic ontologies. Access to the ontologies is provided via a single stand alone server. OWL DL Inferencing and SPARQL based querying are supported.

Data warehousing [31] provides a framework for reporting and analysis tasks across disparate enterprise data systems, for decision support. In data warehousing meta data is modelled to extract information from business systems into a centralised data warehouse. The warehouse is used to create reports and analyse business performance. Data warehouse models can be arranged in normalised form, following the relational approach [32], or ‘dimensional form [31] as quantifiable *facts* and *dimensions* which denote the context of facts. Data warehousing relies on fixed data models and structures, which is in contrast to the more general approach taken in ISO11179, where models are expected to change over time.

The Common Warehouse Metamodel (CWM) [33] from the Object Management Group is a UML based framework to enable data warehousing in practice. However, as with data warehousing, CWM is focused on write-once models, and for working with rigidly structured data. The models and data warehouse structure in CWM are not intended to change after creation and data is copied into the data warehouse from separate systems. The standard consists of 22 parts and the core meta model has overlap with the the *concept* and *value* elements of the ISO11179 meta model.

VII. Conclusion

A. *Ongoing work*

[1]

References

[1] OMG. Unified modelling language. [Online]. Available: http://www.uml.org//

[2] Eclipse. (2008) Ecore tools. [Online]. Available: http://eclipse.org/ecoretools/

[3] Object Modelling Group. Object modelling group. [Online]. Available: http://www.omg.org/

[4] ——. Model driven architecture. [Online]. Available: http://www.omg.org/mda/

[5] National Cancer Intelligence Network. Cancer outcomes and services dataset. [Online]. Available: http://www.ncin.org.uk/collecting_and_using_data/data_collection/cosd/

[6] C. Crichton, J. Davies, J. Gibbons, S. Harris, A. Tsui, and J. Brenton, "Metadata-driven software for clinical trials," in *Proceedings of the 2009 ICSE Workshop on Software Engineering in Health Care*. IEEE Computer Society, 2009, pp. 1–11.

[7] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[8] International Standards Organisation. International Standard for Metadata Registries. [Online]. Available: http://metadata-standards.org/19763/

[9] D. Abler, C. Crichton, J. Davies, S. Harris, and J. Welch, "Models for forms," *Proceedings of the 11th Workshop on Domain-Specific Modeling*, October 2011. [Online]. Available: http://www.dsmforum.org/events/DSM11/

[10] University of Washington. Catalyst web tools. [Online]. Available: http://catalyst.washington.edu/web_tools/index.html

[11] OpenClinica. [Online]. Available: http://www.openclinica.com

[12] 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.

[13] E. K. Nelson, B. Piehler, J. Eckels, A. Rauch, M. Bellew, P. Hussey, S. Ramsay, C. Nathe, K. Lum, K. Krouse *et al.*, "LabKey Server: An open source platform for scientific data integration, analysis and collaboration," *BMC bioinformatics*, vol. 12, no. 1, p. 71, 2011.

[14] P. Fearn and F. Sculli, "The CAISIS research data system," in *Biomedical Informatics for Cancer Research*. Springer, 2010, pp. 215–225.

[15] 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, pp. S103–S108, 2011.

[16] H. Leroux, S. McBride, and S. Gibson, "On selecting a clinical trial management system for large scale, multi-centre, multi-modal clinical research study," *Stud Health Technol Inform*, vol. 168, pp. 89–95, 2011.

[17] P. R. Payne, "Biomedical knowledge integration," *PLoS Computational Biology*, vol. 8, no. 12, 2012.

[18] A. Khambati, J. Grundy, J. Warren, and J. Hosking, "Model-driven development of mobile personal health care applications," in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2008, pp. 467–470.

[19] J. Davies, J. Gibbons, J. Welch, and E. Crichton, "Model-driven engineering of information systems: 10 years and 1000 versions," *Science of Computer Programming*, vol. 89B, pp. 88–104, September 2014.

[20] H. Schlieter, M. Burwitz, O. Schönherr, and M. Benedict, "Towards model driven architecture in health care information system development," in *12th International Conference on Wirtschaftsinformatik (WI 2015)*, March 2015.

[21] Model driven heath tools. [Online]. Available: https://www.projects.openhealthtools.org/sf/projects/mdht/

[22] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. S. Shvo, "HL7 clinical document architecture, release 2," *Journal of the American Medical Informatics Association*, vol. 13, no. 1, pp. 30–39, 2006.

[23] Eclipse MDT UML2 tools. [Online]. Available: https://eclipse.org/modeling/mdt?project=uml2

[24] P. Scott and R. Worden, "Semantic mapping to simplify deployment of HL7 v3 Clinical Document Architecture," *Journal of Biomedical Informatics*, vol. 45, no. 4, pp. 697–702, 2012.

[25] A. A. Sinaci and G. B. L. Erturkmen, "A federated semantic metadata registry framework for enabling interoperability across clinical research and care domains," *Journal of Biomedical Informatics*, vol. 46, no. 5, pp. 784 – 794, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046413000750

[26] S. Jeong, H. H. Kim, Y. R. Park, and J. H. Kim, "Clinical Data Element Ontology for Unified Indexing and Retrieval of Data Elements across Multiple Metadata Registries," *Healthc Inform Res*, vol. 20, no. 4, pp. 295–303, Oct 2014.

[27] C. Tao, G. Jiang, W. Wei, H. R. Solbrig, and C. G. Chute, "Towards Semantic-Web Based Representation and Harmonization of Standard Meta-data Models for Clinical Studies," *AMIA Jt Summits Transl Sci Proc*, vol. 2011, pp. 59–63, 2011.

[28] Linked open vocabularies. [Online]. Available: http://lov.okfn.org/dataset/lov/

[29] Apache stanbol. [Online]. Available: https://stanbol.apache.org/docs/trunk/components/ontologymanager/

[30] Kaon2. [Online]. Available: http://kaon2.semanticweb.org/

[31] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[32] W. H. Inmon, *Building the Data Warehouse*. Wellesley, MA, USA: QED Information Sciences, Inc., 1992.

[33] J. Poole, D. Chang, D. Tolbert, and D. Mellor, *Common Warehouse Metamodel Developer's Guide*. John Wiley & Sons, 2003, vol. 24.