Model Driven Engineering for Large-Scale Clinical Research

Jim Davies, Jeremy Gibbons, Adam Milward, 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) [**?**] or Ecore [**?**]. 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

The data model language was implemented using the Eclipse Modeling Framework (EMF) [**?**], with the class and association entities being based upon Ecore [**?**] classes. Additional metaclasses were included to represent attributes as *data elements*, and the range of values permitted as *value domains*. A value domain has an associated *datatype*, specifying representation, with an optional notion of *units*.

Language extensions for specific implementation types were implemented in the same framework. As an example, consider the metamodel for forms shown in Figure 1, which includes an additional class Section alongside the core classes DataClass, DataElement, DataConstraint, and DataAssociation. This class may contain any DataConcept, including Section itself. As a subclass of DataConcept, it can be linked to data concepts in other models. Other metaclasses, including ValueDomain are not shown.

B. Model Catalogue

The model catalogue stores classes, data elements, constraints, and associations only as components of data models. It stores also administrative links between models—recording, for example, the fact that one model is a new version of another—and semantic links between data concepts: classes and elements.

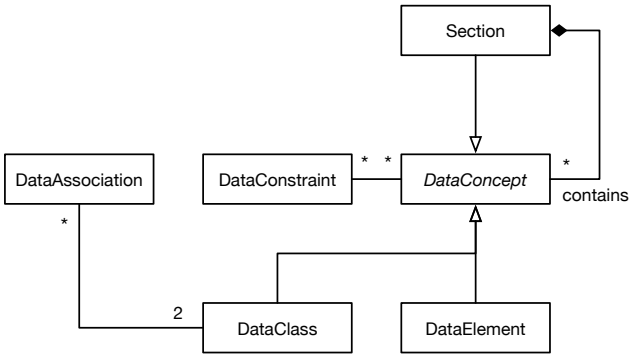

Fig. 1. Extending the data model language

In the current version of the catalogue, associations are not considered as concepts for semantic linking. The instantiation of an association as a reference type is a platform-specific concern, and thus associations are quite different from ordinary data elements or attributes, in having only an abstract notion of a value domain.

At present, if we wish to reason about an association as a concept, we need to introduce an explicit reference or identifier type into our model. This is analogous to the use of foreign keys in a relational database. It would, however, be a relatively straightforward matter to allow linking of associations as abstract data concepts.

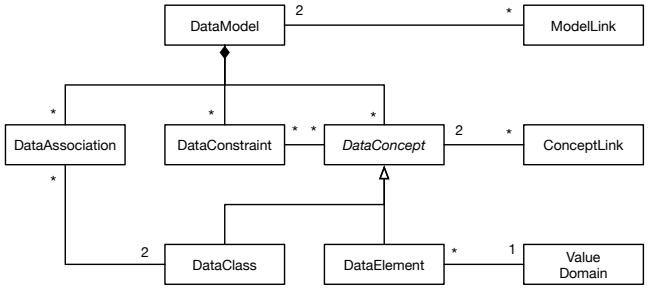

Fig. 2. Model catalogue contents

Figure 2 is a simplified representation of the information stored. The actual implementation includes also the notion of a ValueConcept, which may be used to support semantic linking of value domains and—in the case of an enumerated value domain—the value elements they contain.

C. Metadata registration

In designing the language, we gave due consideration to the ISO/IEC 11179 international standard for metadata registration. This standard sets out a metamodel for data definitions, together with a set of processes for their registration and publication: the core set of metaclasses is shown in Figure 3.

In the ISO/IEC 11179 standard, data elements correspond to individual data points and may be grouped or classified into 'data element concepts'. Each concept represents a set

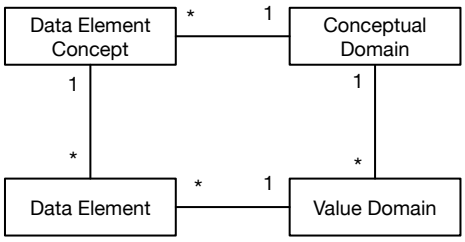

Fig. 3. ISO/IEC 11179 core metaclasses

of data elements that are capturing 'conceptually' the same information: for example, different ways of measuring blood pressure. Value domains are classified into 'conceptual domains', representing different ways of recording the same kind of observations.

The only other form of semantic link supported by the standard is the classification of data element concepts according a single property-based hierarchy. A single classification scheme is unlikely to capture semantic information in a way that suits the purposes of different users, and represents a problem for registration: who is to decide quite where a new data element fits within the existing hierarchy?

The approach taken in our implementation of the model catalogue is more general: the use of semantic links supports multiple classification schemes, and allows the represent of relationships other than simple taxonomies. However, it should be clear that our catalogue could be used, under suitable constraints, as a compliant implementation of the ISO/IEC 11179 standard.

This applies also to the processes of registration, versioning, and publication. Every object stored in the catalogue is managed as an *administered item*, in the language of the standard. The richer notion of linking in the catalogue implementation allows us to exploit this administrative information in the automatic creation and maintenance of semantic links, and the administrative processes are generalised to provide support for collaborative development, but the option of a constrained, compliant implementation remains.

D. Technology platform

The existing model catalogue is built using the Groovy/-Grails framework, which takes a model-view-controller approach to data management and presentation. The key advantage of this platform has been the ability to revisit the underlying data representation—the domain model—without needing to re-implement the presentation layer, and vice versa. As the software was developed in the course of application, this was particularly important.

The plug-in architecture of the framework allowed us to add in cross-cutting features—such as global search, access control, and a collaboration environment—without re-engineering the data representation and presentation aspects. This architecture, and the built-in support for domain-specific langauges, was particularly helpful in the development of model import and export pipelines: facilitating the generation and configuration of software artefacts from models, as well as the generation of models for existing artefacts.

The web interface was developed using the AngularJS JavaScript library, which worked well in support of the continuing interation of the design through interaction and collaboration with users. Although the users were highly-skilled clinical scientists, they had little or no intuition or experience in data modelling. Concepts such as inheritance, reference, composition, and instantiation were unfamiliar and confusing to them. Only through close collaboration were we able to develop a presentation of these concepts that they found accessible.

The persistence layer was implemented using the Grails Object Relational Mapping (GORM), enhancing portability but creating some concerns with regard to scalability: no performance issues have been encountered to date, but the potential exponential increase in complexity, as further linked models are added, and further relationships are inferred, means that an alternative platform may need to be considered.

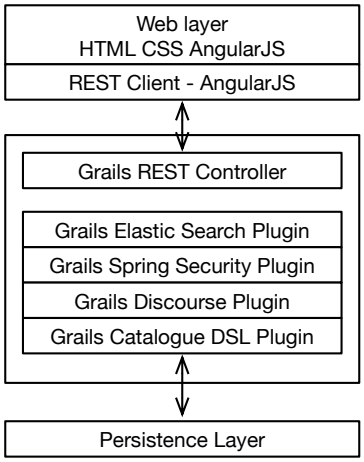

Fig. 4. Implementation architecture

The current technology stack is shown in Figure 4. The discourse plugin provides support for collaborative development of models and data definitions, with users able to contribute to a comment history for each administered item, prompting responses from other users as necessary. This proved particularly important given that many of the clinical scientists were contributing to the dataset development in their spare time.

E. Generation pipelines

The existing implementation has been used to generate several different types of artefact, including:

a) Case report form models for consumption by the OpenClinica clinical trials management system: These take the form of Excel spreadsheets with columns specifying form structure, question text, response types, logical constraints (including skip logic), and presentation controls. These models

are generated from form models in the data modelling language by way of a complex transformation: the hierarchical structure of the data model is flattened to produce lists of sections, repeating groups, and questions. Default values and implementations are included as part of the transformation: for example, we provide custom validation for textual fields that are tagged with constraints in the form of regular expressions.

b) Database triggers: To support the automatic processing of data received from the clinical trials system, we require a collection of triggers for the underlying database. These ensure that the combination of existing and newly-received data is properly normalised. This is particularly important where data is being collected against different versions of the same form.

c) XML schemas for electronic document submission: This is a more straightforward transformation, as the structure of the XML schemas is closer to that of the data models. However, additional processing is required to produce normalised, readable schemas. For example, if there are several data elements sharing the same value domain, we would wish to include that value domain only once within the schema.

d) Tools for creating and validating .csv files: For some of the systems that we are working with, the easiest way to import or export information is in comma-separated value format. In this case, we are not generating a specification of the data format in some implementation language; we are instead generating tools that will ensure that the values presented in a file comply with the model constraints.

e) Data manuals: Datasets and data standards in health informatics are communicated through documents in which each data point is listed along with its intended interpretation. By generating these manuals automatically, we are able to ensure consistency between the information that they present and the tools used for data acquisition and processing. The models used to generate the manual are semantically linked to those used for XML schema and case report form generation.

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 data, and to provide access to genomic medicine across the whole of the country.

The results of the whole genome sequencing will be linked to detailed information on each participant: clinical and laboratory information drawn from health records, ontological statements regarding abnormal features or conditions, and additional information obtained from the participant or their representatives. The information required will depend upon the nature of the disease that the patient is suffering from. For example, information on breast density is required in the case of breast cancer, but not for other diseases.

131 different diseases have been included in the sequencing programme thus far. Each disease corresponds to a different combination of clinical and laboratory data points, a different set of ontological statements, and a different set of questions for the participant.

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 5.

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.

- 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

[1-2] [3-4] 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 5.

During the Genomics England project, which is ongoing, the Models Catalogue toolkit has also 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, register it as a forum subject and invite other clinicians to comment on it. The result is easier coordination and development of the datasets.

Two initial tasks in the GE where to assemble datamodels for Cancer and for Rare Diseases. Although these models are still being developed we can report back the facts that to date the new datamodels have been built with a significant amount of re-use as illustrated in the following four tables.

These figures are a snap-shop of the work in progress, however they are able to illustrate the rate of data element re-use, and of value domain datatype re-use. Both DataModels

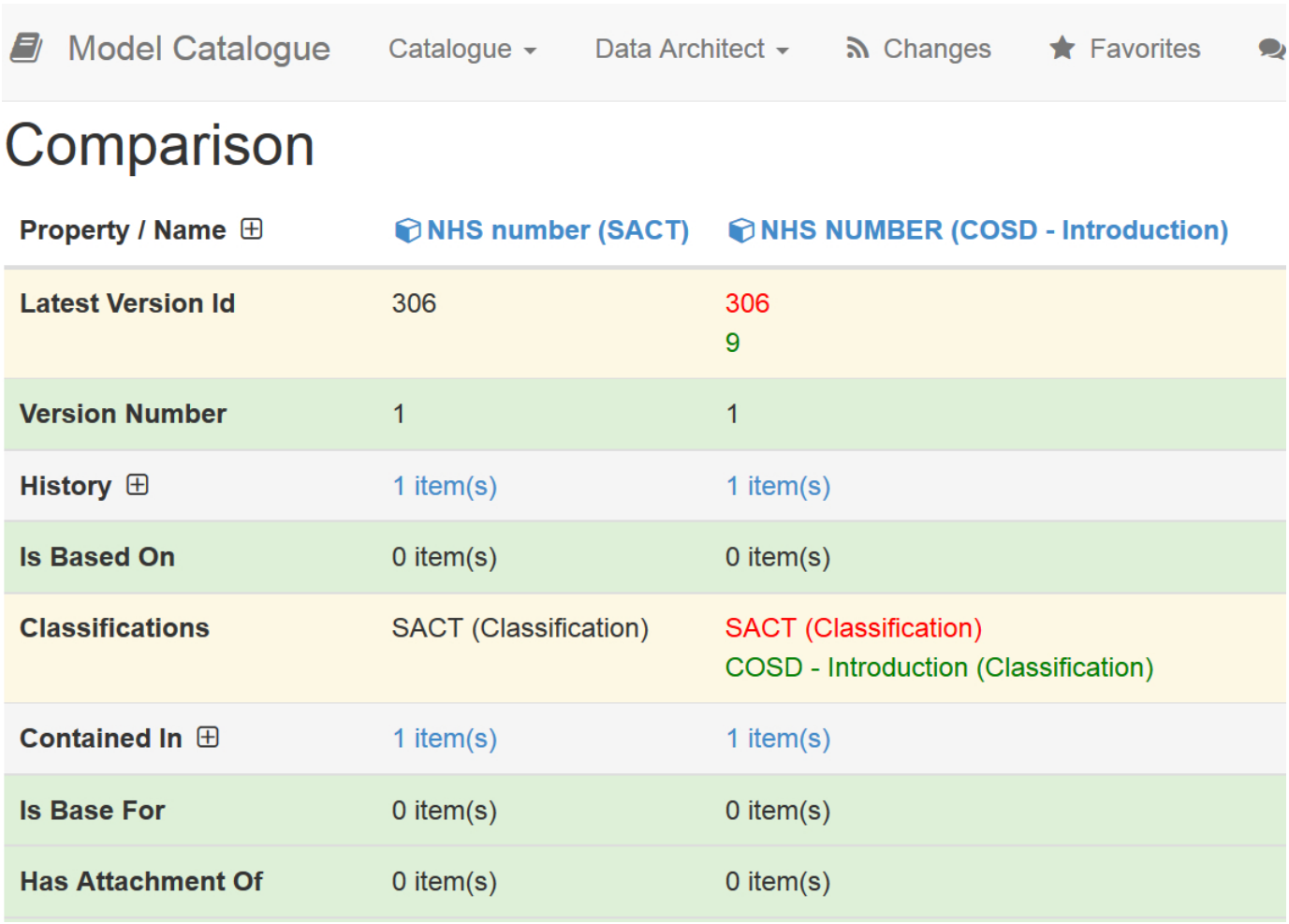

Fig. 5. Screen Shot of DataClass Comparison

TABLE I
CANCER DATA MODEL : DATA-ELEMENT RE-USE

DataModel Name	Elements	%
COSD	60	28.45 %
NHS Data Dictionary	12	5.7 %
SACT	2	0.95 %
New	137	64.9 %
TOTAL	211	

TABLE II
CANCER DATA MODEL : VALUE DOMAIN / DATATYPE RE-USE

Source	Value Domains	%
NHS Standards	28	20.44
XSD Standard datatypes	62	45.26
New datatypes	47	34.31
TOTAL	137	

TABLE III
RARE DISEASES DATA MODEL : DATA-ELEMENT RE-USE

DataModel Name	Elements	%
NHIC	41	10.96 %
NHS Data Dictionary	46	12.3 %
New	287	76.74 %
TOTAL	374	

TABLE IV
RARE DISEASES DATA MODEL : VALUE DOMAIN / DATATYPE RE-USE

Source	Value Domains	%
NHS Standards	81	28.22
XSD Standard datatypes	143	49.83
New datatypes	63	21.95
TOTAL	137	

show an above 20 % re-use rate for data elements, rising to above 65 % for the datatypes. The results are significant and indicate that the re-use of both data elements and value types is significant part of the development of disease datamodels, having a model driven toolkit which can assemble a new datamodel and compare it in detail has clearly benefited the clinicians who are now using it. We admit that the results are not the result of any experiment we have set up to prove the value of the toolkit, but they do strongly support the value of applying model-driven engineering principles in this domain. The 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. Unique IRI's identifying each datamodel, dataclass, dataelement and valuedomain mean that searching for entities is fast, comparison between entities is clear and unambiguous, and generating output which conforms to agreed standards becomes straightforward.

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 [**?**], OpenClinica [**?**], REDCap [**?**], LabKey [**?**] and Caisis [**?**]. A two-year case-study comparing EDC tools is available in [**?**], a further study comparing Clinical Trials Management Systems is found in [**?**]. 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) [**?**] 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 [**?**], [**?**], 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 [**?**], which advocates a multi-phase approach, where data modelling is a separate phase from stakeholder engagement and data integration. This approach is taken in [**?**], 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 [**?**], 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 [**?**], 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) [**?**] project, the HL7 Clinical Document Architecture (CDA) standard [**?**] for managing patient records is implemented using Eclipse UML tools [**?**]. 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 [**?**], 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.

The current technique has evolved from the CancerGrid project [**?**], where an ISO1179-compliant metadata registry was developed for curation of semantic meta data and model-driven generation of trial-specific software [**?**], [**?**]. The practical approach to generating forms in the CancerGrid project has been made more formal by use of an ISO11179 based metamodel, more flexible by use of a plugin architecture and more scalable by use of a collaborative on-line interface, in the current incarnation. In techniques such as [**?**], specific schemas recognised as a key component to enable large scale processing of genomic data, however the approach does not integrate ISO11179 principles and instead uses a single fixed schema for all data. Another effort to develop an implementation of ISO11179 is found in the US caBIG initiative [**?**], however the approach requires a proprietary storage database and the caCORE software development kit [**?**] applies model driven development to generate stubs of web service, requiring developers to create application logic by-hand, whereas our technique integrates with existing clinical Electronic Data Capture tools and workflows, such as OpenClinica [**?**].

Several efforts have addressed the representation of ISO11179 as ontological models for enabling data integration across metadata registries (MDRs). In [**?**] 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 [**?**] 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 [**?**] 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 is based on a manual encoding. In [**?**], existing ontologies are used to enriche OpenClinica forms; the current technique can integrate these ontologies to capture compliant data in a similar fashion, but does so by using an ISO11179 meta data registry and model driven methodology.

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 [**?**] 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 [**?**] 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 [**?**] 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.

In data warehousing [**?**] meta data is modelled to copy information from business systems into a centralised data warehouse, for decision support and to analyse business performance. Data warehouse models can be arranged in

normalised form, following the relational approach [**?**], or ‘dimensional form [**?**] as quantifiable *facts* and *dimensions* which denote the context of facts. The Common Warehouse Metamodel (CWM) [**?**] from the Object Management Group is a UML based framework to enable data warehousing in practice. Data warehousing is focused on write-once models, and for working with rigidly structured data, which is in contrast to the more general approach taken in ISO11179, where models are expected to evolve and change over time. The CWM 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