Fifth IEEE International Conference on Software Engineering and Formal Methods

Model-driven architecture for cancer research

Radu Calinescu, Steve Harris, Jeremy Gibbons and Jim Davies
Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Igor Toujilov and Sylvia B. Nagl
Oncology Department, The Royal Free and University College Medical School
Hampstead Campus, Rowland Hill Street, Hampstead, London NW3 2PF, UK

Abstract

It is a common phenomenon for research projects to collect and analyse valuable data using ad-hoc information systems. These costly-to-build systems are often composed of incompatible variants of the same modules, and record data in ways that prevent any meaningful result analysis across similar projects. We present a framework that uses a combination of formal methods, model-driven development and service-oriented architecture (SOA) technologies to automate the generation of data management systems for cancer clinical trial research, an area particularly affected by these problems. The SOA solution generated by the framework is based on an information model of a cancer clinical trial, and comprises components for both the collection and analysis of cancer research data, within and across clinical trial boundaries. While primarily targeted at cancer research, our approach is readily applicable to other areas for which a similar information model is available.

1. Introduction

This paper presents a framework for the generation of data management software for cancer clinical trials. The framework was implemented as part of CancerGrid [4], a project developing open-standards cancer informatics, and uses a combination of formal methods, model-driven development and service-oriented technologies to automate the generation of software systems for the collection, management and analysis of cancer clinical trial data.

Our overall approach to *trial management system* development is depicted in Figure 1. The clinical trial metamodel underlying the architecture [16] is built as an embodiment of the Consolidating Standards of Reporting Trials (CONSORT) statement [1, 25], the *de-facto* set of guidance rules for the reporting and execution of clinical trials. A trial designer tool provides oncology clinicians with an integrated environment for the design of cancer clinical trials that are instances of the metamodel. This use of a common metamodel for the design of different clinical trials enables the model-driven development of reusable software components for cancer research. Additionally, the tool enforces the consistent use of controlled cancer vocabulary and common data elements (i.e., *cancer metadata*)—a key factor for sharing data across clinical trials. Starting from a set of *trial designs* produced by means of this tool, an assembly of software artefact generators build the complete code for the services, electronic forms and documentation that compose a trial management system. In this paper we present the architecture of our model-driven framework for the generation of data management software for cancer research. The other elements of our approach are described in detail in [5, 16, 29, 30].

The availability of a concise and unambiguous specification was deemed essential for a joint project involving multi-disciplinary teams located at several research institutions across the UK. The Z formal specification language [32] was selected to construct this formal description of the framework based on our prior experience with using Z in software engineering projects. The resources invested in building a formal specification of our architecture paid off, as the resulting Z model has been of great help in gaining and conveying a good understanding of the framework. In particular, the specification of the processes and semantics involved in generating the IT system for clinical trial execution was very useful in defining parts of the architecture that are not captured in the clinical trial metamodel itself, and would have been difficult to specify unambiguously otherwise. Furthermore, the Z specification was used to guide the actual development of software components that automate the generation of clinical trial data management systems, contributing significantly to their timely delivery.

A service-oriented architecture (SOA) was chosen as the

0-7695-2884-8/07 $25.00 © 2007 IEEE
DOI 10.1109/SEFM.2007.18

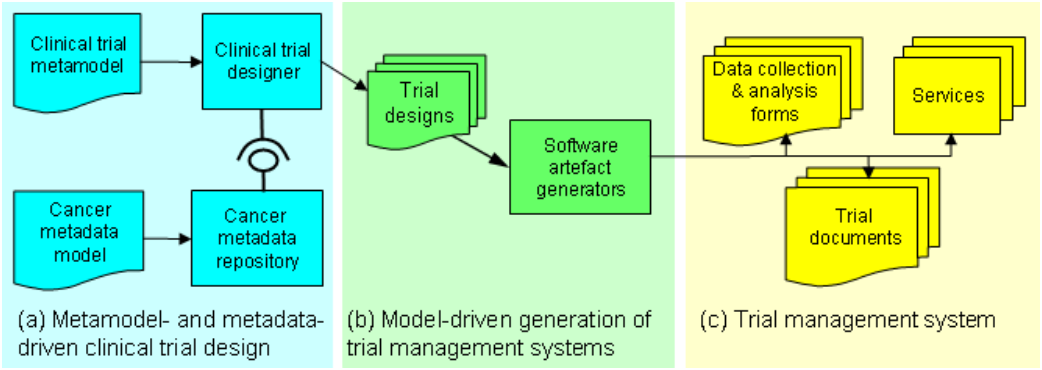

Figure 1. Metamodel- and metadata-based generation of cancer data management systems

target for this generation due to its ability to combine independent, reusable components as required by each specific trial and by the cross-trial data analysis [5]. Another factor that recommended this approach was the good match between the cancer research use cases and the decoupling of services and clients characteristic to SOAs [33].

The remainder of the paper is organised as follows. After a review of related work in the next section, Sections 3 and 4 introduce our common data element and trial design specifications, respectively. Section 5 presents several types of queries involved in the analysis of research data across multiple clinical trials. The techniques used for the model-driven generation of a trial management system are formally defined in Section 6. Section 7 describes the components of the framework and how they co-operate to build an instance of the trial management system. A case study that illustrates the use of the framework for two of CancerGrid's primary clinical trials [26, 27] is provided in Section 8, followed by our concluding remarks in Section 9.

2. Related work

The US cancer Biomedical Informatics Grid (caBIG) project [19] models clinical trials [22] and has cancer data sharing as one of its primary objectives. Their caCORE software development kit [18] handles the generation of web service stubs for some aspects of cancer research, with the actual logic being manually added to these stubs. Our framework succeeds in generating fully-fledged web services for clinical trial execution automatically by including in its underlying model a comprehensive specification of the targeted clinical trials. Additionally, our model-driven approach to generating a trial management system automates the generation of the web forms that, together with these services, provide a complete SOA solution.

The WSDL .NET tool [24] produces web service and client stubs from WSDL contracts and XML schemas. However, in the absence of any knowledge about the service and client logic, the missing code needs to be added manually. In contrast, the use of a generic clinical trials model and of a formally defined set of transformations makes the CancerGrid framework capable of generating web services and forms that can be readily deployed on to an IIS server.

Many commercial and open-source tools have been developed for the design and generation of XForms from XML documents and schemas. XFormation [23] is a commercial tool providing a graphical environment for XForms design and the ability to generate XForms from XML schemas. Although one of the few mature products providing schema-based XForms generation, XFormation does not expose an API for the runtime creation of forms—an important requirement for the CancerGrid framework, whose users need to build web forms from dynamically generated XML schemas. The same applies to IBM's XML Forms Generator [17], a highly-configurable Eclipse plug-in that generates functional forms with XForms mark-up embedded within an XHTML document from an XML data instance, an XML schema or a WSDL document.

XSLT is often presented [15] as a straightforward means for the generation of XForms from XML schemas. One disadvantage of this approach is the complexity of the resulting XSL document. It is easier to develop and maintain a solution of this level of complexity in a traditional programming language such as Java or C#, which is the approach taken by our project. An important benefit of this approach is the ability to use the same generator for producing different types of forms, e.g., XForms and ASP.NET forms.

In the data analysis area, the query of data from multiple sources has been an important research topic for the last two decades. Generic approaches for querying multiple information sources were proposed [2, 7, 8] that use a model of a problem domain to devise global query systems. The approach in [2] requires the user to build a semantic domain model as well as a model of each database and knowl-

edge base used as an information source. Therefore, this solution is appropriate only for users with expertise in both data modelling and the target problem domain. Similar approaches are described in [7, 8], where sophisticated techniques are used to create a "reference data model" [7] or a "metadatabase" [8] that are then employed to generate the global query. Unlike these approaches to querying heterogeneous data sources, our data analysis takes advantage of the homogeneity of data across cancer clinical trials to hide most of the complexity of a cross-trial query. Implementations of this system can therefore be used directly by cancer researchers with limited data modelling expertise.

Other medical projects such as VOTES [28] and PRATA [9] are concerned with the integration of data from multiple, distributed databases. VOTES [28] is interested in the integration of distributed medical data pertaining to the same patient, so candidate patients for new clinical trials can be identified easily. The query forms used by the VOTES portal resemble those from the prototype implementation of the trials management system introduced in this paper, however they are encoded manually by software developers familiar with the internal structure of the data sources. The PRATA system [9] addresses the XML integration of data extracted from multiple, distributed databases. The integration and visualisation of the data is based on a user-specified XML schema that requires inside knowledge of the data sources. In contrast, our data analysis is model-based, and provides information to guide user querying rather than relying on the users for this knowledge.

3. Common data elements

The consistent use of a controlled vocabulary (i.e., a set of domain-specific terms managed by a vocabulary registration authority) is key to sharing data between projects in any field of research. This is particularly relevant to cancer research, where tremendous human and financial resources are employed for the generation of small amounts of data [4]. The ability to analyse these data across multiple clinical trials is crucial to reaching statistically relevant conclusions.

The CancerGrid project is addressing this important requirement by basing its clinical trials model [16] on the use of thesauri – collections of controlled vocabulary terms and their relationships, and common data elements – controlled sets of cancer concepts and measurements. A common data element [29] is defined in terms of several basic types:

- $CdeID$, the set of common data element identifiers used to refer uniquely to specific CDEs;
- $CdeType$, the set of types that common data element values may have;
- $CdeInfo$, the metadata that fully define the semantics of the common data element.

These basic types are summarised below using Z notation:

$$[CdeID, CdeType, CdeInfo],$$

and the common data element type is specified by

$$\begin{array}{|l} \textit{Cde} \\ \hline id : CdeID \\ valueDomain : CdeType \\ info : CdeInfo \\ \hline \end{array}$$

The common data elements for a specific domain are maintained in a CDE (or *metadata*) repository for that area of research:

$$\begin{array}{|l} \textit{CdeRepository} \\ \hline cdeSet : \mathbb{P}\, Cde \\ \hline \forall x, y : cdeSet \bullet x.id = y.id \Rightarrow x = y \\ \hline \end{array}$$

4. Trial designs

Clinical trial data are generated during the execution of a trial as a result of a number of *trial events*, each of which corresponds to a stage in the execution of the clinical trial. For instance, clinical and personal patient data are collected during the registration stage, treatments are allocated in the randomisation stage, and periodical follow-up data collection is performed to assess response to treatment. The complete set of trial events in the CancerGrid trial model is given below:

$$\begin{aligned} TrialEvent ::= {} & registration \mid eligibility \mid randomisation \mid \\ & onStudy \mid treatment \mid offStudy \mid followUp \mid adverseEvent \end{aligned}$$

Clinicians gather the data corresponding to the trial events by filling in *case report forms* that comprise CDEs drawn from the cancer CDE repository [29],

$$\mid cancerCdeRep : CdeRepository.$$

A case report form is fully defined by the sequence of trial events corresponding to its sections:[1]

$$\begin{array}{|l} \textit{CaseReportForm} \\ \hline events : \mathrm{seq}\ TrialEvent \\ \hline \end{array}$$

For the purpose of our model-driven trial system development, a clinical trial is composed of a set of case report forms and the CDEs corresponding to each of their events:

$$\begin{array}{|l} \textit{TrialDesign} \\ \hline forms : \mathbb{P}\, CaseReportForm \\ eventCdeSet : TrialEvent \pfun \mathbb{P}\, cancerCdeRep.cdeSet \\ \hline \mathrm{dom}\, eventCdeSet = \bigcup \{f : forms \bullet \mathrm{ran}\, f.events\} \\ \forall f_1, f_2 : forms \bullet f_1 \neq f_2 \Rightarrow \mathrm{ran}\, f_1.events \cap \mathrm{ran}\, f_2.events = \emptyset \\ \hline \end{array}$$

[1] The set of event handlers for the events in a case report form must be executed in a well-defined order, so it is essential that the trial events associated with the form are ordered instead of being organised into a set.

Two additional operators are used in our specification. The first operator returns the set of events associated with a trial:

$$trialEvents : TrialDesign \rightarrow \mathbb{P}\ TrialEvent$$
$$\forall\, t : TrialDesign \bullet$$
$$trialEvents\ t = \bigcup\{f : t.forms \bullet \operatorname{ran} f.events\},$$

and the second operator builds the set of CDEs associated with a case report form in a clinical trial by combining the CDE sets corresponding to its events:

$$formCdeSet : TrialDesign \times CaseReportForm \nrightarrow \mathbb{P}\ Cde$$
$$\operatorname{dom} formCdeSet = \{t : TrialDesign;$$
$$f : CaseReportForm \mid f \in t.forms \bullet (t, f)\}$$
$$\forall\, t : TrialDesign \bullet \forall\, f : t.forms \bullet formCdeSet\,(t, f) =$$
$$\bigcup\{e : \operatorname{ran} f.events \bullet t.eventCdeSet\ e\}.$$

To give an example of a trial design, consider the CDEs

$$NodalStatus, ECOGStatus, TumorResectionStatus,$$
$$RadiotheraphyTiming, QualityOfLifeSubstudyConsent,$$
$$DiseaseStage, OestrogenStatus, ... : cancerCdeRep.cdeSet$$

that are associated with trial events for the tAnGo trial [27]:

$$tAnGo : TrialDesign$$
$$\exists f : tAnGo.forms \bullet$$
$$f.events = \langle registration, eligibility, randomisation\rangle$$
$$tAnGo.eventCdeSet\ registration =$$
$$\{QualityOfLifeSubstudyConsent, OestrogenStatus, ...\}$$
$$tAnGo.eventCdeSet\ eligibility =$$
$$\{TumorResectionStatus, DiseaseStage, ...\}$$
$$tAnGo.eventCdeSet\ randomisation =$$
$$\{NodalStatus, ECOGStatus, ...\}$$

The common data elements used to register tAnGo participants, to establish their eligibility, and to stratify the allocation of treatments (i.e., the trial *randomisation* [3]) are explicitly specified in the tAnGo trial design. Note that the complete trial design comprises all of the trial events defined at the beginning of this section, however for the sake of brevity only three of these are presented above.

5. Trial data analysis

The *TrialDesign* specification introduced in Section 4 provides all the necessary details for the collection of research data during the execution phase of a clinical trial. In this section, we formally define the data queries that form the basis for the analysis of the trial results. In their most general form, these trial queries are designed to operate on results pertaining to several cancer research projects based on CDEs drawn from the same metadata repository; however we will also demonstrate how specific instances of the queries support result analysis within a single project.

5.1. Event-based data query

Our data queries are built using a number of pre-defined comparison operators:

$$ComparisonOp ::= hasAnyValue \mid isEqualTo \mid$$
$$isNotEqualTo \mid isLessThan \mid isGreaterThan \mid ...$$

Each $CdeType$ type that CDEs can draw their values from is associated with a well-defined set of these operators:

$$cdeComparisonOp : CdeType \rightarrow \mathbb{P}\ ComparisonOp.$$

For our purpose, a query element comprises a CDE and a comparison operator that is relevant for its value domain:

QueryElement
$$cde : Cde$$
$$op : ComparisonOp$$
$$op \in cdeComparisonOp\ cde.valueDomain$$

The query system described in this section is event based, that is, we are interested in identifying CDEs that are associated with the same trial event in all clinical trials involved in the query. This approach is consistent with the cancer researchers' need to analyse data from patients with similar characteristics at the same stage of their treatment. For instance, it makes sense to group patient data collected prior to the commencement of treatment for trials where the treatment varies, because this will avoid the confounding effects of the different treatments under study. However, comparisons at later time points may prove less useful.

Given a set of clinical trials $trialSet$, the query system specifies its query terms as a mapping from trial events to sets of query elements:

TrialQuery
$$trialSet : \mathbb{P}\ TrialDesign$$
$$queryTerms : TrialEvent \nrightarrow \mathbb{P}\ QueryElement$$
$$\operatorname{dom} queryTerms = \bigcap\{t : trialSet \bullet trialEvents\ t\}$$
$$\forall\, e : \operatorname{dom} queryTerms \bullet \{qt : queryTerms\ e \bullet qt.cde\} =$$
$$\bigcap\{t : trialSet \bullet t.eventCdeSet\ e\}$$
$$\forall\, e : \operatorname{dom} queryTerms \bullet \forall\, qt_1, qt_2 : queryTerms\ e \bullet$$
$$qt_1 \neq qt_2 \Rightarrow qt_1.cde \neq qt_2.cde$$

The first constraint in the query definition requires that the trial events involved in the query must be part of all considered trials.[2] The last two constraints state that the query terms for these events comprise precisely one $QueryElement$[3] for every CDE that the event is associated with in each of the queried trials. Notice that

[2] This simplifying assumption is relaxed in Section 5.2, which proposes a generalisation of the event-based cross-trial query.

[3] Although each $queryTerms$ component consists of a single $QueryElement$, the query model can be extended easily to handle terms defined as logical expressions of multiple $QueryElement$s that refer to the same CDE.

the $TrialQuery$ specification does not place any restriction on the comparison operators that are part of the $QueryElement$ items associated with trial events. The only such constraint is specified by the definition of a $QueryElement$, i.e., these operators must be appropriate for the CDEs they relate to.

To illustrate the application of the trial query, consider the NEAT clinical trial [26]:

$$
\begin{array}{|l}
Neat : TrialDesign \\
\hline
\exists f : Neat.forms \bullet \\
f.events = \langle registration, eligibility, randomisation \rangle \\
Neat.eventCdeSet\ registration = \\
\{QualityOfLifeSubstudyConsent, ECOGStatus \\
\quad OestrogenStatus, ...\} \\
Neat.eventCdeSet\ eligibility = \\
\{DiseaseStage, TumorResectionStatus, ...\} \\
Neat.eventCdeSet\ randomisation = \\
\{NodalStatus, RadiotheraphyTiming\}
\end{array}
$$

An event-based query across the tAnGo trial defined in Section 4 and the NEAT trial above is then given by:

$$
\begin{array}{|l}
tAnGoNeatQuery : TrialQuery \\
\hline
tAnGoNeatQuery.trialSet = \{tAnGo, Neat\}.
\end{array}
$$

According to the definition of a $TrialQuery$,

$$
\{registration, eligibility, randomisation\} \subseteq \mathrm{dom}\ tAnGoNeatQuery.queryTerms
$$

since all these trial events appear in both $tAnGo$ and $Neat$. The CDE sets that are part of the $queryTerms$ for the three trial events are:

$$
\begin{array}{l}
\{qt : tAnGoNeatQuery.queryTerms\ randomisation \bullet qt.cde\} \\
\quad = \{NodalStatus\} \\
\{qt : tAnGoNeatQuery.queryTerms\ eligibility \bullet qt.cde\} \\
\quad = \{TumorResectionStatus, DiseaseStage, ...\} \\
\{qt : tAnGoNeatQuery.queryTerms\ registration \bullet qt.cde\} \\
\quad = \{QualityOfLifeSubstudyConsent, OestrogenStatus\}.
\end{array}
$$

Appropriate comparison operators are associated with each of these CDEs in the above $TrialQuery$ instance, e.g.,

$$
\begin{array}{l}
tAnGoNeatQuery.queryTerms\ randomisation = \\
\qquad \{(\!| cde \rightsquigarrow NodalStatus, op \rightsquigarrow isEqualTo |\!)\}
\end{array}
$$

Prior to being executed, a $TrialQuery$ instance such as $tAnGoNeatQuery$ needs to be parameterised by a set of values from the value domains of all CDEs in the query terms. For the $tAnGoNeatQuery$ query term above for instance, op was chosen to be $isEqualTo$, so the $NodalStatus$ value of interest will need to be specified in an implementation of the query framework. This part of the query is not modelled here; however details are provided when a case study illustrating the use of the model-driven generation framework is presented in Section 8.

5.2. Combined-event query

Common data elements used in different clinical trials are not necessarily associated with the same trial event in each of these trials. An example is the $ECOGStatus$ CDE defined in Section 4, which is used by both trials considered in the previous sections, but is associated with different events in the two trials—$ECOGStatus \in tAnGo.eventCdeSet\ randomisation$ and $ECOGStatus \in Neat.eventCdeSet\ registration$. In both cases the measurement is taken prior to patients joining the clinical trial, so the ability to perform queries including query terms for such CDEs is very important.

Having analysed several options for generalising the event-based trial query [6], we opted for an approach that is both simple and effective, and which addresses the use case described above. Our generalised cross-trial query is very similar to the event-based trial query, except that event sets rather than individual events are mapped on to query elements:

$$
\begin{array}{|l}
GeneralisedTrialQuery \\
\hline
trialSet : \mathbb{P}\ TrialDesign \\
queryTerms : \mathbb{P}\ TrialEvent \nrightarrow \mathbb{P}\ QueryElement \\
\hline
\bigcup(\mathrm{dom}\ queryTerms) = \bigcap\{t : trialSet \bullet trialEvents\ t\} \\
\forall E : \mathrm{dom}\ queryTerms \bullet \{qt : queryTerms\ E \bullet qt.cde\} = \\
\quad \bigcap\{t : trialSet \bullet \bigcup\{e : E \bullet t.eventCdeSet\ e\}\} \\
\forall E : \mathrm{dom}\ queryTerms \bullet \forall qt_1, qt_2 : queryTerms\ E \bullet \\
\quad qt_1 \neq qt_2 \Rightarrow qt_1.cde \neq qt_2.cde
\end{array}
$$

The change from individual events to event sets in the definition of the $queryTerms$ component of the query led to a couple of changes to the constraints. The first change relaxes the constraint on the domain of $queryTerms$, specifying it as any set of event sets whose union gives the whole set of overlapping events for the considered trials. The change to the second constraint specifies that the CDE components of the query elements associated with a trial event set are obtained by considering the CDEs associated with all the events in the set.

5.3. Controlled-access query

The data analysis system needs to ensure that access to confidential information is limited to the rightful users. Clinical trials use a role-based access control (RBAC) approach [12, 13] to constrain data access to users that have certain roles in the trial, and our data analysis specification can be further extended to reflect this requirement. A detailed description of this extension is presented in [6], where we prove that the controlled-access trial query associated with a set of trial designs is equivalent to the standard trial query based on the trial design set obtained by eliminating all inaccessible CDEs from the original trial designs.

6. Implied processes and semantics

The clinical trials model covers only part of what is required to architect a complete SOA system. Therefore, the model-based generation of trial management systems also relies on implicit knowledge about cancer clinical trial processes and semantics. Notice that in addition to exposing the clinical trial data model, the $TrialDesign$ schema specifies the process decomposition of trial executions into a set of case report form submission processes. Similarly, the trial event sequence in a case report form defines the basic event handler processes composing each case report form submission.

Given the high-level description of a trial design, the generation of the clinical trial SOA is based on three basic types—XML schemas with a single top-level element (termed *simple XML schemas*), web forms, and web method fragments (i.e., blocks of code that can be combined to form a web service method):

$$[SimpleXmlSchema, WebForm, WebMethodFragment].$$

A web service is therefore defined as a set of web methods, each of which is a sequence (i.e., an ordered collection) of web method fragments:

$$\begin{array}{|l} WebService \\ \hline methods : \mathbb{P}(\text{seq } WebMethodFragment). \\ \hline \end{array}$$

For brevity, the properties that make a sequence of web method fragments into a valid web method, and a set of web methods into a valid web service are not given here. The trial SOA component generation relies on the implicit domain knowledge defined by the following techniques:

1. A schema derivation method that produces the simple XML schema associated with a set of CDEs,

   $$schemaDerivation : \mathbb{P}\ Cde \rightarrow SimpleXmlSchema$$

   The method consists in pulling together all the CDE definitions in the CDE set, and wrapping them within a root XML element definition. Notice that this simple procedure is not specific to cancer research, but can be applied to any research field that uses CDE sets for data representation.

2. A technique for building the web form associated with a simple XML schema, and a web method fragment for handling the submission of the form,

   $$\begin{array}{l} submissionForm : SimpleXmlSchema \rightarrow WebForm \\ submissionHandler : SimpleXmlSchema \rightarrow \\ \qquad WebMethodFragment. \end{array}$$

   The web method fragment created by the CancerGrid generation framework validates the submitted form against the schema (reporting any errors to the client) and logs the form and its validation result. Again, these techniques are independent of the problem domain.

3. Techniques for generating the web services for handling individual trial events, and the web method fragments for calling their methods:

   $$\begin{array}{l} eventHandlingService : TrialEvent \times \mathbb{P}\ Cde \rightarrow \\ \qquad WebService \\ eventSubmissionHandler : TrialEvent \times \mathbb{P}\ Cde \rightarrow \\ \qquad WebMethodFragment \end{array}$$

   The former technique is used for the generation of web services dedicated to special trial events such as $randomisation$ or $adverseEvent$ (and their associated CDEs), while the latter technique is used to create the code that initiates calls to these services when the submission of a case report form is handled.The two mappings are specific to cancer clinical trials, although similar mappings are likely to exist for other problem domains.

4. Techniques for generating web services and forms for data analysis across a set of clinical trials:

   $$\begin{array}{l} analysisService : \mathbb{P}\ TrialQuery \nrightarrow WebService \\ analysisForm : \mathbb{P}\ TrialQuery \nrightarrow WebForm \\ \hline \text{dom } analysisService = \text{dom } analysisForm = \\ \quad \{trialSet : \mathbb{P}\ TrialDesign \bullet \{query : TrialQuery \mid \\ \qquad query.trialSet = trialSet \bullet query\}\} \end{array}$$

   These techniques are also applicable to any research area characterised by an information model similar to that used by our framework.

The techniques presented above are first used to build the *case reporting service* for a clinical trial, i.e., the service handling the submission of all the case report forms in the trial. This service comprises a web method for each case report form in the trial. Each such web method is obtained by concatenating the web method fragment corresponding to the $submissionHandler$ and the fragments that encode the calls to the event-specific services for all form events:

$$\begin{array}{l} caseReportingService : TrialDesign \rightarrow WebService \\ \hline \forall\, t : TrialDesign \bullet (caseReportingService\ t).methods = \\ \quad \{f : t.forms;\ s : \text{seq } WebMethodFragment \mid \\ \quad \text{dom } s = \text{dom } f.events \wedge \\ \quad (\forall\, i : \text{dom } s \bullet s\ i = eventSubmissionHandler\ (f.events\ i, \\ \qquad t.eventCdeSet\ (f.events\ i))) \bullet \langle submissionHandler\ (\\ \qquad\quad schemaDerivation\ (formCdeSet\ (t, f)))\rangle \frown s\} \end{array}$$

With a service-oriented architecture defined simply as a combination of web forms and web services,

$$\begin{array}{|l} Soa \\ \hline webForms : \mathbb{P}\ WebForm \\ webServices : \mathbb{P}\ WebService \\ \hline \end{array}$$

the CancerGrid framework generates the trial management system below:

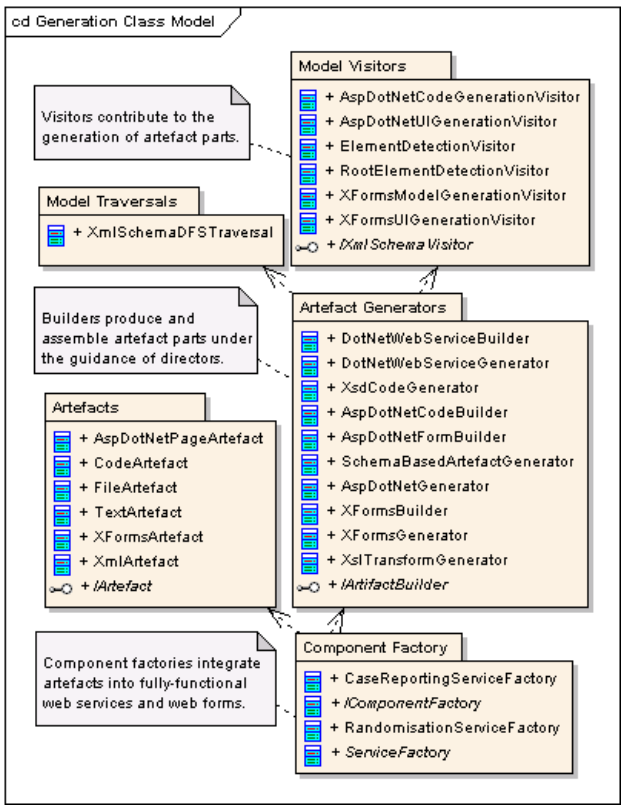

Figure 2. SOA generation framework

$$
\begin{array}{|l}
trialManagementSystem : \mathbb{P}\ TrialDesign \rightarrow Soa \\
\hline
\forall\ T : \mathbb{P}\ TrialDesign \bullet \\
(trialManagementSystem\ T).webForms = \\
\quad \{t : T;\ f : CaseReportForm \mid f \in t.forms \bullet \\
\quad submissionForm\ (schemaDerivation\ (formCdeSet\ (t, f)))\} \\
\quad \cup \{analysisForm\ \{q : TrialQuery \mid q.trialSet = T \bullet q\}\} \\
\wedge\ (trialManagementSystem\ T).webServices = \\
\quad \{t : T \bullet caseReportingService\ t\} \cup \\
\quad \{t : T;\ event : TrialEvent \mid event \in trialEvents\ t \bullet \\
\quad eventHandlingService\ (event, t.eventCdeSet\ event)\} \cup \\
\quad \{analysisService\ \{q : TrialQuery \mid q.trialSet = T \bullet q\}\}.
\end{array}
$$

As a result, the trial management system comprises: the web forms for all case report forms in the trial set and for the cross-trial data analysis; per-trial case reporting service instances; all the event-specific services; and the data analysis service for the considered set of trials.

7. The SOA generation framework

The CancerGrid SOA generation framework (Figure 2) consists of several parts. The *Component Factory* produces the *Artefacts* (i.e., the code and data elements) that compose the clinical trial SOA modules by employing of a number of *Artefact Generators*. In turn, the generators use *Model Traversals* to apply specialised *Model Visitors* to fragments of the trial design, as described in detail below.

SOA components and artefacts for cancer clinical trials The trial management system comprises a set of interdependent web services and web forms. Given the software platforms available to most CancerGrid users, the amount of generated code required for the same functionality, and the available development tools, .NET was chosen over J2EE as the target platform for the web service implementation, and a combination of ASP.NET [11] and XForms [10] is used for the implementation of web forms. Several classes of artefacts are used to build these components:

- *TextArtefact*—A simple, run-time generated string.
- *FileArtefact*—A file on the local file system, e.g., a template configuration file that a generator will include in the components it creates. Typically, file artefacts are independent of the trial for which they are used.
- *XmlArtefact*—An XML document, typically trial-specific and produced at run time by a generators.
- *AspDotNetPageArtefact*—An XML document representing an ASP.NET form.
- *CodeArtefact*—A generated code module.
- *XFormsArtefact*—A trial-specific XForms artefact.

Common to all these artefacts is their implementation of an *IArtefact* interface, whose single method permits artefacts to be saved to an output stream.

Model traversal This class library provides the model traversal support required during artefact generation. In the current version of the system, its only component is a depth-first-search (DFS) traversal of XML schemas represented as (.NET) Schema Object Models (SOMs). Potential candidates for new versions of the generation toolset include traversal of XMI-encoded [21] UML models. The DFS schema traversal provides simple methods for the traversal of schemas, schema sets and of specific elements within schemas. Each such method is using a visitor that does the actual work of generating artefacts from the model.

Model visitors This part of the framework comprises classes that implement the web form and web service fragment generation techniques from Section 6. The implementation uses the hierarchical visitor design pattern [31] (i.e., a variant of the visitor pattern in [14]) to build web forms and web service fragments based on XML schemas derived from a trial design. Each type of visitor generates an artefact or contributes a part to the generation of an artefact:

- *RootElementDetectionVisitor* identifies the single root element of a simple XML schema.

- *ElementDetectionVisitor* finds a pre-specified element from an XML schema.
- *XFormsModelGeneratorVisitor* and *XFormsUIGenerationVisitor* are used to generate the model and user interface parts of XForms [10], respectively.
- *AspDotNetUIGenerationVisitor* is used to generate the UI of ASP.NET forms, and *AspDotNetCodeGenerationVisitor* generates the data binding code required to create the object for the form submission.[4] These visitors implement the $submissionForm$ transformation from Section 6.

Visitors implement the *IXmlSchemaVisitor* interface, which specifies the operations that visitors must support. This includes methods to be employed when entering and exiting each type of non-leaf node encountered during the traversal of a schema object (i.e., *VisitEnter* and *VisitLeave*), and simple *Visit* methods that are used when a leaf node in the schema is reached. The boolean value returned by each of these methods indicates the direction in which the traversal should continue after the method returns. Thus, the children of a non-leaf node are visited if and only if the *VisitEnter* method called for the node itself returns $true$. Similarly, the value returned by a *VisitExit* or *Visit* method specifies whether the next sibling in the model will be visited—returned value $true$, or not—returned value $false$.

Artefact builders and component generation This module (Figure 2) comprises elements that generate individual artefacts from a trial design fragment. Typical examples include .NET web service `.asmx` and *code behind* files, or ASP.NET web forms generated from XML schemas. Generators employ specialised artefact part *builders* to produce an artefact, i.e., they play the *director* role from the builder software design pattern [14]. The framework comprises generators for web service code (*DotNetWebserviceGenerator*), ASP.NET form UI and code (*AspDotNetGenerator*), XForms model and UI (*XFormsGenerator*), data types from XML schema (*XsdCodeGenerator*) and XSL transformations (*XslTransformGenerator*).

SOA component factories These classes generate system components such as web services and forms using techniques from our Z specification. Each component factory bases its operation on several trial design *fragments* obtained by filtering the trial design by means of a set of XSL transforms. These filters represent a combination of the $eventCdeSet$ mapping from trial events to CDE sets, and the $schemaDerivation$ conversion of CDE sets to XML schemas, as described in the Z specification in Section 6.

[4] A limitation of ASP.NET is that data binding is one way: web form fields can get their values from the fields of an associated object, but the reverse is not supported.

The *case reporting service factory* is a specialised service factory that is responsible for generating the case reporting web service and forms required in a clinical trial. The factory uses a "case reporting schema" design filter to extract the information required to build its case reporting artefacts. The result of applying this filter to the trial design is an XML schema whose top-level elements define the case report forms in the clinical trial. This schema is used to generate the data types for the case report forms exchanged between the web forms and the case reporting service web methods, as well as to build the web service and the associated web forms. The $submissionHandler$ and $eventSubmissionHandler$ transforms from the specification in Section 6 are used for this generation.

The *event-handling service factories* implement the $eventHandlingService$ transform (Section 6) to generate the web services for handling specific trial events such as $eligibility$, $randomisation$ or $adverseEvent$. The design filters used extract the CDE sets associated with individual events by means of the $eventCdeSet$ mapping from the $TrialDesign$ Z schema. For instance, the randomisation service factory generates the patient randomisation service for a clinical trial. Two design filters are used to gather the information required to generate the randomisation service. The former filter generates an XML schema for the subset of case report form fields associated with the randomisation trial event. The latter is a "stratification" filter that extracts trial design information about the way in which the value domains of the stratification CDEs are partitioned into subdomains for the purpose of treatment allocation.

Work is underway to add a *data analysis service factory* and the visitors on which it relies to the framework. This factory will employ the $analysisService$ and $analysisForm$ transforms from Section 6 to generate the data analysis form and service, a step in the generation of the trial management system that is currently handled manually.

8. Case study

In order to assess the effectiveness of our framework, we applied it to two real cancer clinical trials that completed their data acquisition and are currently in the analysis stage [26, 27]. This enabled us to devise simulated executions of the two trials based on made-up but realistic patient data, and to demonstrate them to clinical trial personnel directly involved in running the two trials. The feedback obtained from clinicians, statisticians and IT staff that took part in the execution of the trials was then used to improve the specification of the framework, and the general usefulness of the trial management system it generates.

For each of the considered clinical trials, we started from an English-text description of the trial termed a *trial protocol*. Among other information, trial protocols contain de-

tailed descriptions of the data collection and analysis stages of the associated trials. Based on these descriptions, we built instances of our clinical trial metamodel (i.e., trial designs) that are equivalent to the original trial protocols. The resulting $tAnGo$ and $Neat$ trial designs (presented in Sections 4 and 5, respectively) were then used as input for the trial management system generators described in the previous section, which produced the system artefacts—web services and web forms—ready for deployment.

We deployed and configured the automatically generated case reporting and trial event handling services, and the case report forms for the two clinical trials. The data analysis form and service described by the Z specification in Sections 5 and 6 were implemented manually, as the current version of the system does not include generators for these artefacts.

By combining demonstrations of our architecture with a high-level presentation of the associated specification, we managed to convey a good understanding of our approach to a representative sample of its intended users. As a result, valuable comments and suggestions for improvement have been obtained from these users even from the early stages of the project. The ability to speed-up trial management system development and the enforced usage of controlled cancer metadata were consistently deemed as the key advantages of our approach. The data analysis components, although not automatically generated yet, were also perceived as powerful tools for analysing cancer research data both within and across clinical trial boundaries. Patient workflow was reported as insufficiently supported by the current version of the framework, and we are in the process of adding it to the Z specification in the first instance. Based on past experience, we expect this to form the basis for agreeing on the required new functionality with the users of the IT system generated by our framework.

9. Conclusions

The model-driven development framework described in this paper formally defines and automates the creation of the web services and forms for trial management systems in cancer research. Starting from a *design* of a clinical trial, the current version of the framework employs software component generators to produce the artefacts (i.e., the code, data and configuration files) associated with a clinical trial. The successful generation of fully operational SOA components for a couple of clinical trials indicates that automating the model-driven generation of trial management systems is a viable approach.

The use of a formal specification was essential to defining our architecture in a concise and unambiguous manner. The resulting Z model expresses not only the cancer trial metamodel on which the architecture is based, but also the internal processes involved in the generation of the software artefacts that compose a trial management system. Although the work to use the specification in the verification of the generators in our architecture and the software artefacts they produce is not complete yet, the specification has proved invaluable in improving our understanding of the framework and in describing it to other members of the project and to many of its intended users.

Additional work is carried out within the project to develop generators for the data analysis components whose specification is given in the paper, and to add automatic packaging, configuration and deployment to the actual component generation. Other extensions that are being investigated include the separation of the clinical trial logic into standalone BPEL workflows [20] to support the patient workflow associated with clinical trials, and the generation of the SOA security components. These extensions are being developed at the same time as a new, enhanced version of the CancerGrid clinical trials model [16].

While the association of particular events and event handlers with well-defined data sets is inherently domain specific, the largest part of the model-driven development approach is not directly related to cancer research. This part can be readily applied to other areas for which a CDE-based information model of research projects is available. In particular, the techniques used to generate the web forms for data collection and analysis, and the services handling data submission and data queries, respectively are immediately applicable to other types of clinical trials, and to other research fields.

Acknowledgements

This work was supported by the UK Medical Research Council grant G0300648 and a Microsoft Research grant. The authors are grateful to Charlie Crichton, Peter Maccallum, Andrew Tsui and the other members of the CancerGrid team for many insightful discussions during the work on the model-driven development framework.

References

[1] D. G. Altman, K. F. Schulz, D. Moher, M. Egger, F. Davidoff, D. Elbourne, P. C. Gotzsche, and T. Lang. The revised CONSORT statement for reporting randomized trials: Explanation and elaboration. *Annals of Internal Medicine*, 134(8):663–694, April 2001. http://www.annals.org/cgi/reprint/134/8/663.pdf.

[2] Y. Arens, C. Y. Chee, C.-N. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, June 1993.

[3] E. Beller, V. Gebski, and A. Keech. Randomisation in clinical trials. *Medical Journal of Australia*, 177:565–567, November 2002. http://www.mja.com.au/public/issues/177_10_181102/bel10697_fm.pdf.

[4] J. Brenton, C. Caldas, J. Davies, S. Harris, and P. Maccallum. CancerGrid: developing open standards for clinical cancer informatics. In *Proceedings of the UK e-science All Hands Meeting 2005*, pages 678–681, 2005. http://www.allhands.org.uk/2005/proceedings/.

[5] R. Calinescu. Model-based SOA generation for cancer clinical trials. In *Best Practices and Methodologies in Service-Oriented Architectures. Proceedings of the 4th OOPSLA International Workshop on SOA and Web Services*, pages 57–71, Portland, October 2006.

[6] R. Calinescu, S. Harris, J. Gibbons, and J. Davies. Cross-trial query system for cancer clinical trials. In *Advances in Systems, Computing Sciences and Software Engineering—CISSE 2006*. Springer, 2007. To appear.

[7] S. Castano, V. D. Antonellis, and S. D. C. di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, March/April 2001.

[8] W. Cheung and C. Hsu. The model-assisted global query system for multiple databases in distributed enterprises. *ACM Transactions on Information Systems*, 14(4):421–470, October 1996.

[9] G. Cong, W. Fan, X. Jia, and S. Ma. PRATA: A system for XML publishing, integration and view maintenance. In *Proceedings of the UK e-Science All Hands Meeting*, pages 432–435, Nottingham, UK, 2006.

[10] M. Dubinko. *XForms Essentials*. O'Reilly, 2003.

[11] B. Evjen, S. Hanselman, F. Muhammad, and S. Sivakumar. *Professional ASP.NET 2.0*. Wiley Publishing, Inc., 2006.

[12] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Computer Security Series. Artech House, 2003.

[13] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, August 2001. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison Wesley, 1995.

[15] E. Gropp. Transforming XML schemas, 2003. http://www.xml.com/pub/a/2003/01/15/transforming-schemas.html.

[16] S. Harris and R. Calinescu. CancerGrid clinical trials model 1.0, 2006. CancerGrid technical report MRC/1.4.1.1, www.cancergrid.org/public/documents.

[17] IBM Corporation. XML forms generator, 2006. http://www.alphaworks.ibm.com/tech/xfg.

[18] US National Cancer Institute. The caCORE Software Development Kit, 2006. http://ncicb.nci.nih.gov/infrastructure/cacoresdk.

[19] US National Cancer Institute. The cancer Biomedical Informatics Grid, 2006. https://cabig.nci.nih.gov/.

[20] M. B. Juric et al. *Business Process Execution Language for Web Services*. Packt Publishing, 2004.

[21] J. Kovse and T. Harder. Generic XMI-based UML model transformations. In Z. Bellahsene, D. Patel, and C. Rolland, editors, *Object-Oriented Information Systems: 8th International OOIS Conference*, volume 2425 of *Lecture Notes in Computer Science*, pages 192–198. Springer, 2002.

[22] R. Kush. Can the protocol be standardised? Technical report, Clinical Data Interchange Standards Consortium, 2006. http://www.cdisc.org/publications/CDISK_ed.pdf.

[23] Focus Software Ltd. XFormation, 2006. http://www.xformation.com/default.asp.

[24] Microsoft. Web services description language tool (wsdl.exe), 2006. http://msdn.microsoft.com/library/en-us/cptools/html/cpgrfwebservicesdescriptionlanguagetoolwsdlexe.asp.

[25] D. Moher, K. F. Schultz, and D. G. Altman. The CONSORT statement: revised recommendations for improving the quality of reports of parallel-group randomised trials. *The Lancet*, 357, April 2001.

[26] C. Poole and H. Earl. NEAT: National breast cancer study of epirubicin plus CMF versus classical CMF adjuvant therapy. http://www.ncrn.org.uk/portfolio/dbase.asp.

[27] C. Poole, H. Howard, and J. Dunn. tAnGo: A phase III randomised trial of gemcitabine in paclitaxel-containing, epirubicin based adjuvant chemotherapy for women with early stage breast cancer, 2003. http://www.isdscotland.org/isd/servlet/FileBuffer?namedFile=tAnGo_protocol_version_2.0_July_2003.pdf.

[28] A. Stell, R. Sinnott, and O. Ajayi. Supporting the clinical trial recruitment process through the grid. In *Proceedings of the UK e-Science All Hands Meeting*, pages 61–68, Nottingham, UK, 2006.

[29] I. V. Toujilov and P. Maccallum. Common data element management architecture. Technical Report MRC-1.1.2, CancerGrid, May 2006. http://www.cancergrid.org/public/documents/2006/mrc/Report%20MRC-1.1.2%20CDE%20management%20architecture.pdf.

[30] I. V. Toujilov and S. B. Nagl. Client's script execution in semantic web services. In *Proc. 10th International Conference on Intelligent Engineering Systems*, pages 247–252, London, UK, 26–28 June, 2006.

[31] Cunningham & Cunningham, Inc. (C2) wiki. Hierarchical visitor pattern. http://c2.com/cgi/wiki?HierarchicalVisitorPattern.

[32] J. Woodcock and J. Davies. *Using Z. Specification, Refinement and Proof*. Prentice Hall, 1996.

[33] O. Zimmermann, P. Krogdahl, and C. Gee. Elements of service-oriented analysis and design, June 2004. http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/.