Form Follows Function

Model-Driven Engineering for Clinical Trials

Jim Davies[1], Jeremy Gibbons[1],
Radu Calinescu[2], Charles Crichton[1], Steve Harris[1], and Andrew Tsui[1]

[1] Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
http://www.cs.ox.ac.uk/firstname.lastname/

[2] Computer Science Research Group, Aston University
Aston Triangle, Birmingham B4 7ET, UK
http://www-users.aston.ac.uk/~calinerc/

Abstract. We argue that, for certain constrained domains, elaborate model transformation technologies—implemented from scratch in general-purpose programming languages—are unnecessary for model-driven engineering; instead, lightweight configuration of commercial off-the-shelf productivity tools suffices. In particular, in the *CancerGrid* project, we have been developing model-driven techniques for the generation of software tools to support clinical trials. A domain metamodel captures the community's best practice in trial design. A scientist authors a trial protocol, modelling their trial by instantiating the metamodel; customized software artifacts to support trial execution are generated automatically from the scientist's model. The metamodel is expressed as an XML Schema, in such a way that it can be *instantiated by completing a form* to generate a conformant XML document. The same process works at a second level for trial execution: among the artifacts generated from the protocol are models of the data to be collected, and the clinician conducting the trial instantiates such models in reporting observations—again by completing a form to create a conformant XML document, representing the data gathered during that observation. Simple standard form management tools are all that is needed. Our approach is applicable to a wide variety of information-modelling domains: not just clinical trials, but also electronic public sector computing, customer relationship management, document workflow, and so on.

It is the pervading law of all things organic and inorganic,
Of all things physical and metaphysical,
Of all things human and all things super-human,
Of all true manifestations of the head,
Of the heart, of the soul,
That the life is recognizable in its expression,
That form ever follows function. This is the law.

Louis Sullivan [33]

1 Introduction

Randomized controlled trials are considered to be the 'gold standard' for experiments in medicine. They provide the most reliable evidence supporting or refuting a scientific hypothesis, such as that 'treatment X cures more patients suffering from disease D than does treatment Y'. An experiment is designed: treatment regimes X and Y will be specified; patients suffering from disease D will be recruited; recruits will be stratified into groups with similar relevant characteristics, based on factors such as age, gender and lifestyle; patients within each group will be allocated at random to treatment X or treatment Y; the results will be analyzed to determine whether or not the difference in effect of the treatments is statistically significant. So far, so good.

From a software point of view, a clinical trial is largely an exercise in data management: observations have to be specified, collected, recorded, integrated, and analyzed. But the software engineering aspects of setting up and running a clinical trial are not trivial. Two particular problems that we will address involve *data integration* and *tool generation*.

The data integration problem occurs because medical researchers want to be able to combine the results of multiple trials, a process known as *meta-analysis*. It is often the case that a single trial in isolation does not have adequate statistical power to yield a robustly significant conclusion. Nevertheless, if sufficiently many trials have been conducted, investigating sufficiently similar hypotheses and collecting sufficiently similar data, it may be possible to pool the results to greater effect: *"... the drug Tamoxifen—an oestrogen blocker that may prevent breast cancer cells growing—was the object of forty-two studies world-wide, of which only four or five had shown significant benefits. But this did not mean that Tamoxifen did not protect against breast cancer. When we put all the studies together it was blindingly obvious that it does..." [34].* In other situations, the meta-analysis aims at evaluating new hypotheses that are formulated long after the completion of the trials that originally collected the data involved—in this case, data from trials investigating quite different hypotheses may be integrated.

Either way, for meta-analysis to be possible, it is necessary to capture and curate metadata expressing the 'semantics' of the data; only then is it possible to determine whether data collected in different trials are commensurate, and if so, how to relate them. For example, when measuring blood pressure, it is not enough to record a pair of numbers, or a pair of pressures, or a pair of measurements in mmHg, or even to indicate that these represent systolic and diastolic pressure; it is also necessary to know how that data was collected (at rest, or after five minutes on a treadmill?), and maybe factors such as who collected it (in the clinic by a professional, or at home by the patient?) and how recent and reliable it is. This semantic metadata is an essential part of the context of the data, and logically forms part of the model of the trial—alongside more syntactic metadata such as the name of the trial and the types of data items.

As for tool development, standard practice in clinical trials management these days is to pass a textual document containing the trial protocol over to database

programmers in the clinical trials unit, or to consultants from a trials management software provider, who will use it as guidance in manually configuring an information management system for this particular trial. This practice causes numerous problems. Firstly, it induces delays: it is usually the case that some to-ing and fro-ing is needed between the database programmers and the medical researchers to get the details right, but the medics are too busy to respond immediately, and the start of the trial is often delayed because the software is not ready. Secondly, it is costly. This is not such a problem for a big 'phase III' trial for measuring effectiveness of a treatment: this will have thousands of participants and a stable design, so the software development will form only a small proportion of the overall cost, and anyway such a trial is likely to be commercially funded rather than run on a shoestring. But it certainly becomes a problem for 'early-phase' exploratory studies, which are much smaller, more dynamic, and poorly funded. Thirdly, it is not uncommon for an early-phase trial protocol to undergo changes during the execution of the trial, requiring adjustments to software components of the associated trial management system; current practice is to implement these changes through manually modifying the underlying code, running the risk of introducing software bugs when the system is in production use. And finally, bespoke database design on a per-trial basis is unlikely to promote the consistency and interoperability needed for meta-analysis.

All four of these generation issues could be addressed if the development of the software tools needed to support trial execution could be automated. Fortunately, there is essentially enough information in the trial protocol—which needs to be written anyway, not least for the purposes of regulatory approval—to completely determine the relevant software artifacts, either from scratch or by configuring more generic components. If the protocol were written in a more structured format—that is, as a formal model, rather than merely a textual description, of the trial—then both the prose and the code could generated from it by suitable processing, and any adjustments required because of changes to the trial protocol can be made without risky manual intervention at the level of code. Moreover, as we have seen, the annotation of the data descriptions in the trial model with semantic metadata will make that model doubly useful, as a basis for supporting meta-analysis in addition to being a specification for a software system.

In other words, clinical trials management is crying out for a model-driven engineering (MDE) approach.

Contribution

Our main contribution in this paper is a demonstration that model-driven engineering need not involve the modeller—who is ideally a domain expert rather than a developer—in 'programming' in a textual or graphical notation; rather, in certain constrained domains, the modelling task is sufficiently formulaic that it can be conducted simply by completing forms. Our argument is illustrated by way of examples drawn from clinical trials management; but we also discuss its applicability to other information domains, such as electronic government. In

this paper we focus on the model-driven aspects of the problem; data integration is discussed in a companion paper [12].

2 The CancerGrid Project

The CancerGrid project [7] was instigated to address the twin problems of interoperability and generativity in clinical trials, taking a model-driven approach to the development of trials management tools. It was funded in the first instance for three years from 2005 by the UK Medical Research Council, with the involvement of five UK universities: Cambridge (specializing in oncology), Oxford (software engineering), University College London (semantic modelling), Birmingham (clinical trials management), and Belfast (telemedicine). Oxford University Software Engineering Programme and the Cancer Research UK Cambridge Research Institute have been continuing the work.

An early activity of the project was the reification of the *Consolidated Standards of Reporting Trials* (CONSORT) statement [27] as a domain model. The CONSORT statement is a widely-adopted checklist of thirty-two items, intended to capture best practice in reporting clinical trials: name, date, sponsor, unique unique trial number, target recruitment size, randomization protocol and stratification variables, eligibility criteria, case report form structure, clinical interventions, and so on. We expressed this checklist as a class model of well-designed trials; this model is described in detail in a technical report [19], and the principles behind the model discussed in a companion paper [12].

An intriguing aspect of the CancerGrid approach is the parallels it reveals between the 'design-time' and 'run-time' stages in the lifecycle of a trial. In the first stage, a scientist is designing a trial protocol, following the guidelines set out in the CONSORT statement; in the second stage, a clinician is conducting a trial, following the guidelines set out in the trial protocol. Both cases involve instantiating a schema: the trial protocol instantiates the CONSORT class model, but it determines schemas for data collection and reporting at significant events during the trial, which are themselves instantiated by the clinician when entering data. We therefore call our model of the CONSORT statement the *metamodel*, because each protocol that instantiates it is itself a model of a particular trial. This parallel is discussed in more detail in Section 4, where the workflow of trial design and execution is presented.

Much of the interoperability requirement pivots on some kind of consensus on—or at least, machine-processable documentation of—the *common data elements* (CDEs) being recorded. There can be no magic here: if two trials have collected incompatible data, or one of them has provided insufficient metadata to determine compatibility, then their results cannot usefully be integrated. On the other hand, it is very difficult to arrange for prior universal agreement on compatible data elements across a large, heterogeneous, and long-lived community.

The approach to this dilemma that we have taken on the CancerGrid project is realist rather than idealist. We have developed tools to support communities

in deciding on, recording, and disseminating data standards; but there is no need for all parties to commit to using the same standard. We have developed the *CancerGrid Metadata Registry* (cgMDR) [8], an open-source implementation of the ISO 11179 metadata registry standard [20]. This is robust enough for widespread use—it is currently being adopted by the US National Cancer Institute (`http://www.cagrid.org/display/MDR/`), for example—while still being lightweight enough for individual trials units to install their own copy to support local practice and customization. Data elements (for example, 'blood pressure on induction into study, measured at rest') are curated in the metadata registry, and referenced in the trial protocol; the metadata reference is preserved in the software artifacts generated from the protocol—data entry forms, database schemas, spreadsheets, and so on—ensuring that all the data maintain their semantic annotations throughout their journey through the system.

As the name suggests, the original plan in 2005 was to use the then-emerging 'grid' technologies as a basis for the implementation. This turned out to be impractical: the toolkits were relatively unsophisticated, requiring considerable programming effort to duplicate the functionality of applications that were already available to our target users; moreover, 'grid computing' in the sense of large-scale computational or storage clusters is not relevant to this particular problem. The development activity was thus refocussed upon the production of software that worked to extend and configure applications that are widely used and available within the UK National Health Service (and, indeed, throughout government and industry): specifically, Microsoft Office and SharePoint Server. The project's focus upon the requirements of clinical researchers, and the recognition that these requirements can be met partly through the (automated) enhancement and configuration of office productivity applications, has led to changes in attitudes. As one team member put it: "We used to be hung up on open source; now we're really focussed on open standards."

Also as suggested by the name, the initial focus was on cancer clinical trials; the first validations were on three breast cancer trials [28, 29, 16]. The tools have also been demonstrated in an Anglo-Canadian cancer clinical study, in which clinical data and tissue sample metadata from five centres were integrated automatically on the basis of declared semantics in cgMDR, allowing an analysis across 4000 samples that would have been impractical under existing approaches; and in a proof-of-concept usage on the US Veterans' Health Administration Cooperative Studies Program, where forms have been generated for serious adverse event reporting in a rheumatoid arthritis study that allow the incorporation of metadata directly from the US National Cancer Institute Thesaurus.

Current applications of the CancerGrid approach include: *Accelerating Cancer Research Using Semantics-Driven Technology* [3], funded by Microsoft Research, exploring the extension from phase III to early-phase studies; *Evolving Health Informatics* [14], funded by Research Councils UK, working with colleagues in the Centre for Clinical Vaccinology and Tropical Medicine at the University of Oxford to demonstrate applicability to infectious disease control; *Hospital of the Future*, aiming to improve patient outcomes through information-

driven management; the *Data Support Service*, funded by the UK Medical Research Council (MRC), to retrospectively catalogue the data collected in some of the MRC's valuable long-running studies; and the *Union of Light-Ion Centres in Europe*, funded by the European Union Seventh Framework Programme, to curate experimental results in particle therapy.

3 Forms-Based Model-Driven Engineering

The Object Management Group's *Model-Driven Architecture* [26] aims to raise the level of abstraction in software development, by separating higher-level specifications of system functionality from lower-level descriptions of the implementation of that functionality on a particular platform. *Platform-independent models* (PIMs) abstract away from technical details of implementation, such as representation in terms of CORBA objects or SOAP-based web services; these technical details only appear in *platform-specific models* (PSMs). The transformations from PIMs to PSMs (and for that matter, between PIMs and between PSMs) might be fully automated, or might require some degree of human intervention.

Models must be expressed in some structured notation, such as UML or XML. The construction of models has a number of benefits over going straight to code. Firstly, a higher-level PIM is, presumably, simpler than a lower-level PSM, being less cluttered with details; this makes it easier to analyse, to test, and to reason about. Secondly, the effort involved in constructing and refining PIMs may be amortized over multiple PSMs, encouraging investment in reusable assets. And thirdly, interoperability between systems can be more clearly specified in terms of PIMs, with integration mechanisms consequently inferred from PSMs. If all one has is code, then all one can do is to execute it; whereas higher-level models may be put to multiple uses.

The approach to model-driven engineering taken in the CancerGrid project is a document-centric one: *data models as document schemas*; *entities as conformant documents*; *authoring as form completion*; and *model transformations as schema mappings*. Moreover, there is a pleasing symmetry between design-time and run-time activities: the former consists of constructing a system model as a document that conforms to a domain metamodel, the latter consists of recording an event as a document that conforms to a data model, and both involve instantiation via form completion. This view is perhaps not a universally appropriate approach to model-driven engineering—more complex modelling notations might not be readily viewed as document formats, and form-filling might not be the most convenient way of authoring models in such notations—but it works in restricted domains, and in particular in clinical trial design.

In a way, we are heading in the opposite direction to that taken by the *Executable UML* camp [23]. Rather than thinking of models as programs written in a high-level programming language, we think of them as plain data; the programs are simple generic interpreters of this data, written once and reused many times. For example, the trial metamodel is manifested as an XML Schema, and the 'trial designer' application is simply an off-the-shelf tool (in our case, Microsoft

InfoPath) that allows a scientist to design a trial by completing a form derived from that schema. In this sense, we are following Brooks' advice [4]: "*Show me your flowchart and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowchart; it'll be obvious*", or, as later pithily paraphrased by Steele [31]: "*Smart data structures and dumb code works a lot better than the other way around.*"

4 Implementation Approaches

The general principles of data models as document schemas, entities as conformant documents, authoring as form completion, and model transformations as schema mappings can be realized in a variety of ways. Our current implementation uses an off-the-shelf product for form-filling—namely Microsoft InfoPath [24], an application that supports the design and completion of XML-based data entry forms, forming part of the Microsoft Office productivity suite. This has turned out to be the approach most attractive to our target audience of medical researchers and clinicians, because they are all familiar with the Microsoft Office interface, and they all have the software pre-installed on their desktop computers. This approach is described in Section 4.1. However, the general principles are in no way tied to this particular realization, and in Section 4.2 we briefly discuss an alternative implementation approach constructed from first principles.

4.1 Off-the-Shelf Implementation

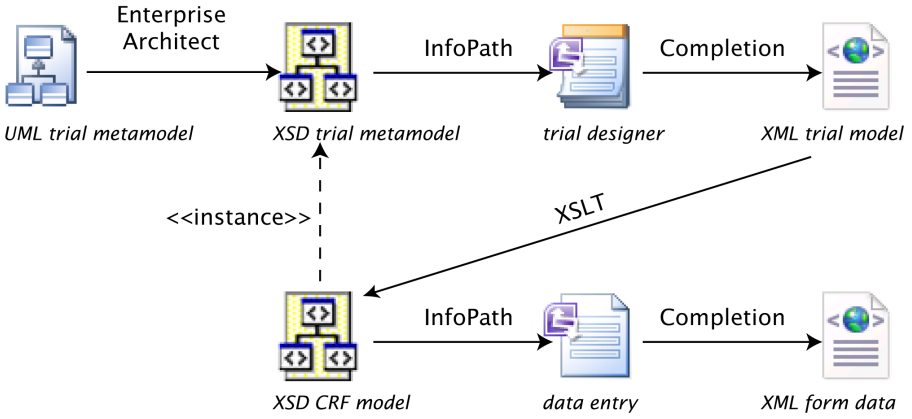

Fig. 1. The forms-based workflow of clinical trial design and execution; icons denote artifacts, and solid arrows denote transformations between artifacts

The workflow entailed by trial design and execution is illustrated in Figure 1. The steps involved are expanded below, and illustrated by means of a paediatric vaccinology study in Kathmandu [13] that we are currently supporting.

First, representatives of the data community design a metamodel of the kinds of experiment in which they are interested, capturing the design in the form of an XML schema; this may be done once and for all, or more realistically, it is subject to relatively infrequent updates. In our case, the general discussion had already taken place in the medical research community, resulting in the existing CONSORT statement [27], which has remained stable for over a decade. We expressed the CONSORT metamodel as a UML class model, and exported the metamodel as an XML schema. This metamodel is quite elaborate, but is partitioned into packages defining trial description, patient eligibility, randomization, treatments, case report forms and form controls, and security policy; a fragment representing form controls is shown in Figure 2. This aspect of the CancerGrid philosophy is described in more detail elsewhere [12]. (In addition, but also once and for all, we developed a number of trial-independent software artifacts, such as generic randomization and case report form management web services. And of course, we are exploiting off-the-shelf applications such as InfoPath and SharePoint.)

Second, a medical researcher planning a clinical trial designs the trial protocol. The *trial designer* application that they use to do so is just InfoPath configured with the trial metamodel. Designing a trial amounts to completing the forms that InfoPath presents; the result is an XML document modelling this particular trial that, by construction, conforms to the schema from the first step—and hence also to the CONSORT statement. (In contrast, CONSORT conformance is not guaranteed with the current practice of writing the protocol in plain English.) Figure 3 shows a screenshot of InfoPath being used to design a case report form for recording study participant registration; the highlighted element is the 'study group' (a three-way enumeration) into which the participant is placed, and a version of the underlying XML representation of this piece of the form, heavily edited for space and readability, is shown in Figure 4(a).

Third, the XML document recording the trial protocol determines numerous software artifacts relating to the trial: clinical interventions, datasets and collection procedures, software configurations for services such as randomization and validation, documentation, and so on. Each trial-specific artifact essentially instantiates the template for such artifacts included in the metamodel—hence the dashed realization arrow in Figure 1 stereotyped '⟪instance⟫', in a slight abuse of notation. (To avoid clutter, the only trial-specific artifact shown in the figure is the model of a case report form (CRF). Other artifacts are also models, but may be models of entities such as services, documents, or workflows, rather than of forms.) The specification of each artifact is obtained by traversing the XML document, extracting the corresponding parts, and transforming them into the appropriate format: XML Schema, XSL Formatting Objects, WSDL, and so on. Traversal, extraction, and transformation is specified as a collection of XSLT stylesheets, written once only, for all trials based on the same version of the metamodel. In particular, the data to be collected by the clinician conducting the trial—and hence the structure of the form on which this data is recorded—is specified by an XML schema. The data manager in the trials unit generates

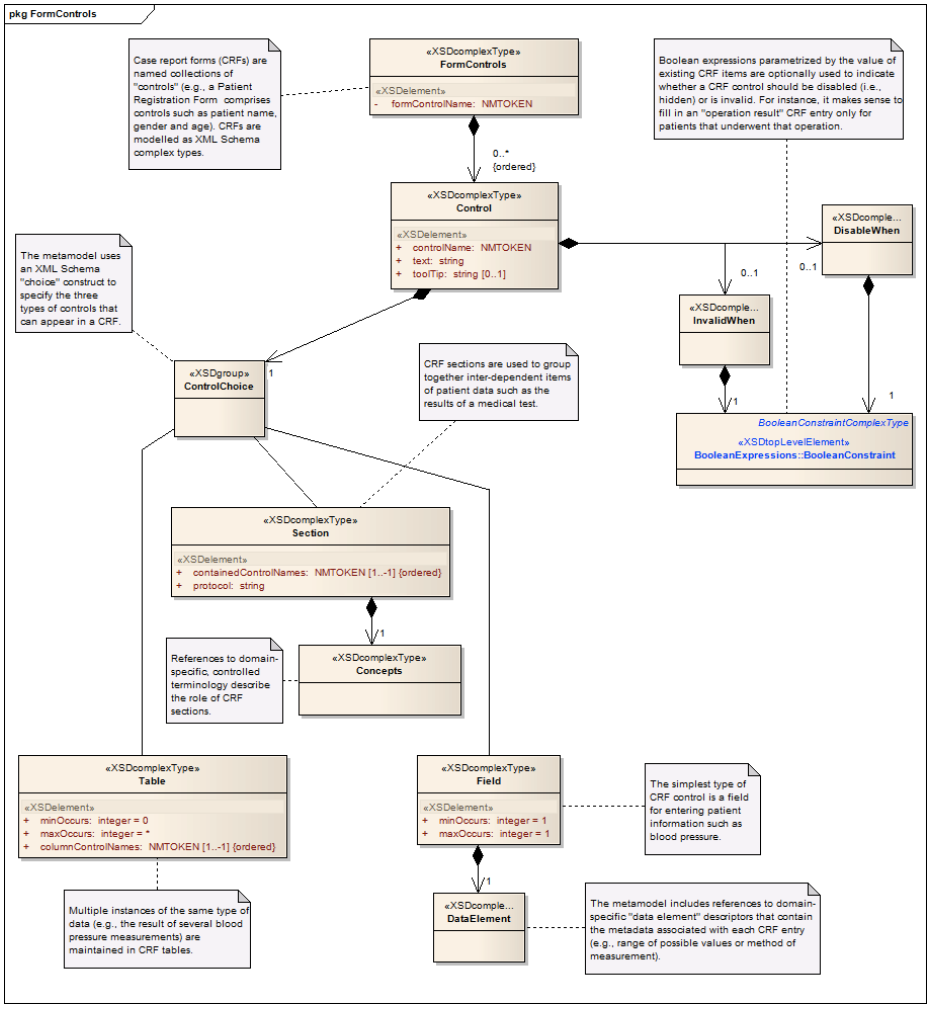

Fig. 2. UML class model of the Form Control package of the CancerGrid metamodel

all these artifacts automatically, and deploys them in the unit's web portal for access by clinicians. Continuing the example, the XML defining the participant study group is used to generate an XML Schema for the data that should be collected; the corresponding portion of that schema, again heavily edited for readability, is shown in Figure 4(b).

Finally, the clinician in the unit running the trial conducts a consultation with a participant in the trial, makes some clinical observations, and needs to record the data so obtained. The *data entry* application that they use to do so is just InfoPath configured with the model of the relevant data. Data entry amounts to completing the form that InfoPath presents; the result is an XML

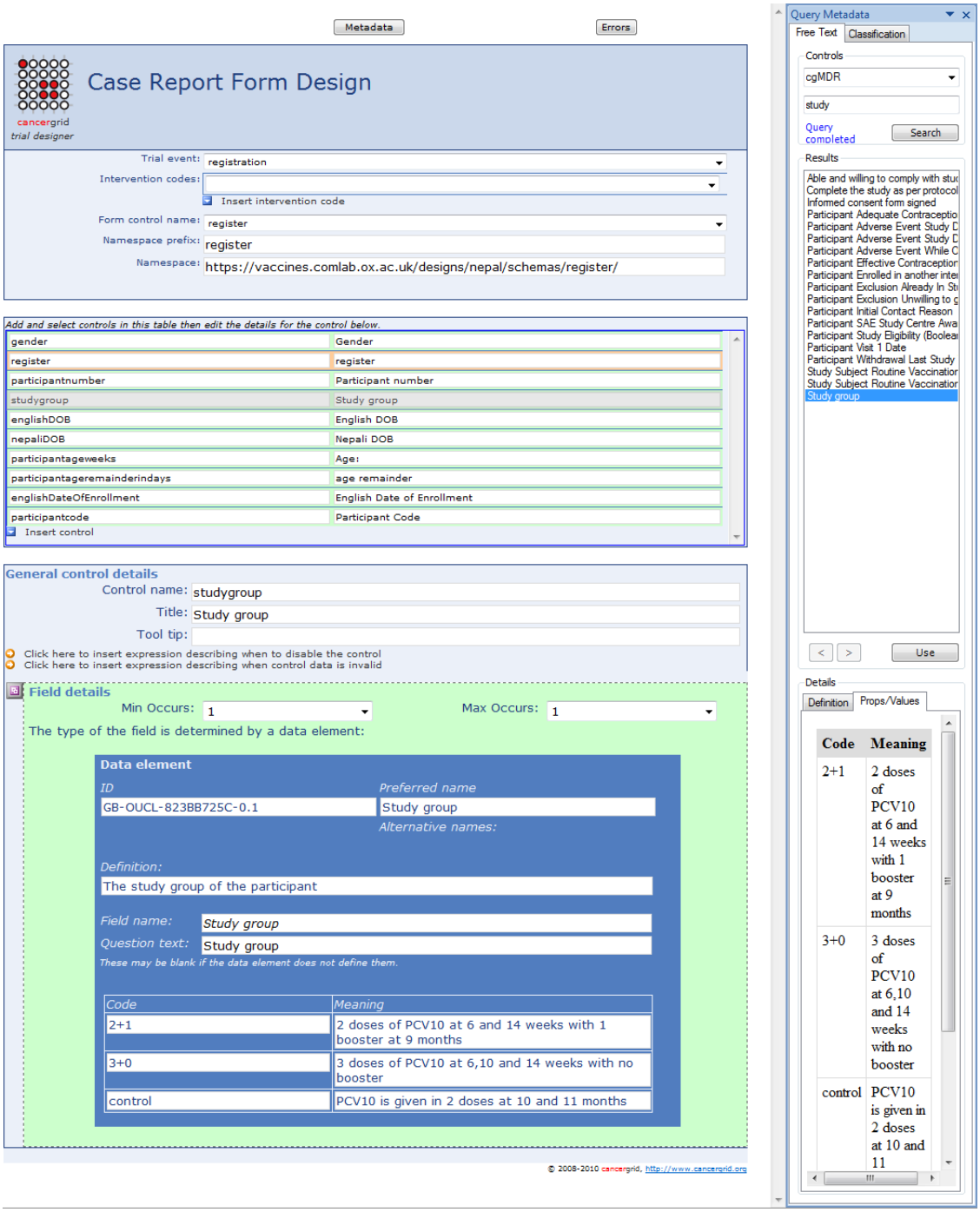

Fig. 3. Using InfoPath to design a case report form

(a)
```
<controlName>studygroup</controlName>
<text>Study group</text>
<minOccurs>1</minOccurs>
<maxOccurs>1</maxOccurs>
<data-element>
  <id>GB-OUCL-823BB725C-0.1</id>
  <definition>The study group of the participant</definition>
  <valid-value>
    <code>2+1</code>
    <meaning>2 doses of PCV10 at 6 and 14 weeks with 1 booster at 9 months</meaning>
  </valid-value>
  <valid-value> ... </valid-value>
</data-element>
```

(b)
```
<xs:element name="studygroup" type="register:SimpleType_studygroup"/>
<xs:simpleType name="SimpleType_studygroup"
               sawsdl:modelReference="GB-OUCL-823BB725C-0.1">
  <xs:annotation><xs:documentation source="definition">
    The study group of the participant
  </xs:documentation></xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="2+1" />
    <xs:enumeration value="3+0" />
    <xs:enumeration value="control" />
  </xs:restriction>
</xs:simpleType>
```

(c)
```
<register:studygroup>3+0</register:studygroup>
```

Fig. 4. Fragments of XML representing (a) a trial model, (b) a data schema derived from the model, and (c) recorded results conforming to the schema

document recording this event that, by construction, conforms to the appropriate schema from the third step, and which may be stored in an XML database for subsequent analysis and study. In the example, choices for the participant study group are presented as a dropdown list; selection from this list results in the creation of the XML in Figure 4(c).

Notice especially that the execution-time actions of the clinician filing a case report closely mirror the design-time actions of the scientist modelling the trial: both actions are manifested as completion of a form to generate a document that conforms to a certain schema. The actors may play different roles in the overall process, but both are domain specialists rather than software professionals, and for both participants a familiar generic application configured with an appropriate model is a suitable tool.

Notice also a significant benefit of the model-driven approach: the model can be used for other purposes than just ‘execution’. For example, as well as the software artifacts derived from a trial protocol, a textual description has to be generated for submission to the relevant regulatory bodies. We have implemented a simple ‘reporting tool’ to generate such a description.

4.2 Implementation from First Principles

The implementation approach described in Section 4.1 is straightforward and robust (on account of being a simple specialization of a proven off-the-shelf framework), and is attractive to our particular end users (on account of their familiarity with this framework). However, a closed commercial off-the-shelf framework may not be appropriate for all environments; so as an exercise in risk mitigation, we also developed a proof-of-concept implementation from first principles. We present a brief outline here; space limitations preclude a more detailed description.

The thoroughness with which the CONSORT statement specifies the requirements for randomized clinical trials enabled us to develop, once and for all, trial-independent software artifacts for managing clinical trial data. We used generic programming techniques to implement these artifacts as .NET web services parametrized by form models. This specialization of the generic trial services using trial-specific form datatypes is analogous to the configuration of Microsoft InfoPath using form schemas.

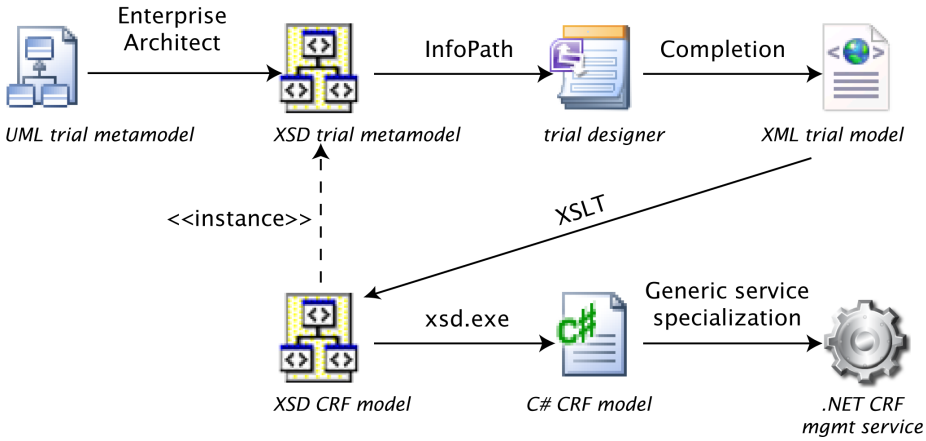

Fig. 5. The forms-based workflow of clinical trial service generation; as in Figure 1, icons denote artifacts, and solid arrows denote intermediate transformations

Figure 5 illustrates the workflow involved in generating CRF management services for a clinical trial. The first part—leading to the generation of an XML schema model of the case report form—is identical to the first part of the workflow presented in Section 4.1 (see Figure 1). For the second part of the service generation workflow, the off-the-shelf XML Schema Definition Tool (Xsd.exe) [25] is used to produce a C# class corresponding to the XML schema model of the case report form—in other words, a C# model of the form. This C# CRF model is then used to specialize the generic CRF management service, and thus to obtain the trial-specific CRF management service.

The same process is entailed by the generation of other similar run-time artifacts, including services for the management of patients, staff (e.g., clinicians, research nurses and statisticians) and locations (i.e., the healthcare research units involved in the clinical trial). Indeed, the same process could be used at design time as well, for authoring the trial protocol: this could use a generic data entry form completion application, analogous to InfoPath but parametrized by a C# class rather than by an XSL schema.

5 Evaluation

The approach to model-driven engineering of clinical trials management software that we have described—representing data models as document schemas, software artifacts as conformant documents, authoring as form completion, and model transformations as schema mappings—has been validated in a number of experiments, as outlined in Section 2. These have included both highly regular phase III trials and more exploratory early-phase studies; both metadata-only and metadata-plus-data exercises; both prospective data capture and retrospective data cataloguing; and both national and international communities. The results are promising, although still on a small scale; we have in gestation some more ambitious experiments for larger-scale validation.

We have been able to demonstrate, for a wide variety of types of clinical study, that bespoke applications, tailored to specific study designs, can be delivered quickly through the automatic configuration of (relatively) low-cost general-purpose software, and that the approach can be validated for the purposes of regulatory submissions. Anecdotal evidence suggests that the reduction in development time may range between 20% and 80%, depending upon the organizational context. Based on the early evaluations mentioned in Section 2, it seems that the reduction in licensing costs can be measured in terms of the cost of a Microsoft Office SharePoint Server select license (of the order of £1000, covering arbitrarily many studies) against that for a typical solution based on commercial domain-specific tools (which "would cost in the range of hundreds of thousands of dollars" [17]).

The major benefit of our forms-based model-driven approach is that it enables the domain expert (i.e., the scientist modelling the trial) to take part in the development of the software, reducing the errors and delays that often occur when informal models are interpreted by an IT expert who lacks domain knowledge. Additionally, both the effort to bring a live software system back into line with a new version of a trial protocol, and the risk of introducing defects in doing so, are eliminated almost completely by automating the process. Another (still unexplored, but nevertheless considered) benefit is the ability to use software instances generated from preliminary protocol versions for testing and iteratively improving the protocol, with minimal development effort; in contrast, current practice delays the identification of flaws in the trial protocol until very late in the development cycle.

6 Conclusions

We have described an approach to model-driven engineering for information-rich domains that can be characterized by a domain metamodel. Model-building is a matter of instantiating the metamodel; when the metamodel is sufficiently regular, that process can reduce to simple form-filling, and can be conducted by domain experts with no programming skills. For many tasks, the same process is applicable a second time when running the application: the model determines a data format, and data gathering again reduces to form-filling, constructing data items that conform to the model. What is most interesting is that modern productivity frameworks—such as the SharePoint and InfoPath components of Microsoft Office—are sufficiently powerful and flexible to support this kind of application, almost without having to resort to low-level implementation in a general-purpose programming language. This is evidence of a raise in the level of abstraction provided by current mainstream software tools.

It has been gratifying to see that the ideas we have developed are much more widely applicable than originally envisaged. Of course, we expected that software engineering efforts related to breast cancer ought to be readily translatable to other types of cancer, and we hoped that its area of applicability would also embrace other diseases; our results with communities working on rheumatoid arthritis and in vaccinology have vindicated that hope.

But we have been pleasantly surprised to learn that essentially the same approach is also highly relevant in the field of electronic government. This too turns out to be largely a problem of data integration: the former UK Prime Minister Tony Blair coined the term 'joined-up government' as a vision for how different government departments ought to—but generally do not at present—interact [1]. Moreover, electronic governance is also a domain in which model-driven generation of software artifacts would be extremely helpful: accountability of public servants requires government information systems to be transparent, and the monopoly typically held by the incumbent government requires the systems to be trustworthy. Our ideas in this area are still under development, but we have some preliminary results [10, 15, 11], and we are discussing further progress with the UK Public Sector Object Model group and with the Scottish Government.

Broadly speaking, we expect the approach to be applicable to any semantically rich domain in which there is: a relatively stable metamodel of the domain; a 'design phase' consisting of instantiating the metamodel to yield a model of a particular instance, which can be used to configure generic software tools; and an 'execution phase' in which entities conforming to the model are created. For example, one could use the approach for a generic conference management system. The basis is a metamodel of academic conferences. The conference chair 'designs' the particular conference by instantiating the metamodel, specifying properties such as whether there is an author response period, whether reviews are double-blinded, how many reviewers each paper should have, and so on. 'Execution' consists of creating entities such as 'submissions' and 'reviews' that

conform to conference-specific aspects of the model. The reader can doubtless think of many similar configurable information-gathering exercises.

One aspect of ongoing work is to extend the scope of the metamodel to cover also the temporal aspects of a clinical trial. Although the trial protocol provides structured specifications of static aspects, in terms of common data elements, the dynamic aspects—when interventions should occur—are described only in free text (see the 'meaning' fields in Figure 3). These too could be specified in a structured format in the protocol, in a workflow modelling notation such as BPEL or BPMN, and then used to generate scheduling tools for trial execution. We have conducted some preliminary studies on using such workflow notations to specify and check trial safety properties such as drug interactions [38, 40], but have not yet integrated this work with the rest of the CancerGrid toolchain. The biggest challenge will be to allow the trial designer to describe the temporal aspects of the trial in sufficient detail, without degenerating into a full-blown programming exercise; we hope that *workflow patterns* [37] and *property specification patterns* [39] will be helpful in this regard.

7 Related Work

The US cancer Biomedical Informatics Grid (caBIG) [36] have a metamodel of clinical trials [21], and have the sharing of cancer clinical data as a primary objective. However, their caCORE software development kit [35] handles only the generation of web service stubs for some aspects of cancer informatics, requiring manual intervention by software developers in order to fill out the logic. Their trial models, in contrast to ours, are not detailed enough to specify the behaviour of these services, and so their generation process can only go as far as an outline rather than a complete implementation.

There are many data interchange formats for medical data, especially for healthcare records—the EN 13606 standard for electronic health record communication (`www.en13606.org`), the Health Level Seven messaging standards (`www.hl7.org`), the CDISC Operational Data Model for interchange of clinical data (`http://www.cdisc.org/odm`), and so on; any of them could be used in conjunction with the CancerGrid metamodel-based tools. Other medical informatics projects are concerned with the integration of data from multiple, distributed databases. For example, VOTES [32] integrates distributed medical data pertaining to a single patient, so that candidate patients for new clinical trials can be identified easily; the query forms used by the VOTES portal resemble those discussed here, but they are encoded manually by software developers familiar with the internal structure of the data sources, rather than being generated from a higher-level model. The PRATA system [9] addresses the XML integration of data extracted from multiple, distributed databases, but based on a user-specified XML schema that requires inside knowledge of the data sources.

Sierra *et al.* [30] describe a document-centric approach to application development, similar to ours in the sense of involving a domain metamodel, models as structured documents, and model-driven generation of software artifacts; but

they expect domain experts to author models in markup languages like XML rather than by simpler form completion. McLaren and Wicks [22] discuss a generative framework using XML; they present an 'indirect' approach using XSLT to turn an XML model into an XML—or other format of—software artifact, and a 'direct' approach using traditional (for example, Java) code to parse and interpret XML on the fly. (Indeed, there is a continuum of approaches; it is really a question of how much of the behaviour may be precomputed, and how much is left until runtime.) The indirect approach is more agile, and is better while requirements are evolving rapidly; but with a large system, the transformation process may take considerable time, and McLaren and Wicks argue that the 'faster direct implementation is necessary once requirements are well defined'. Our approach is at the indirect end of the spectrum, with very little hand-crafted code; even with our two-stage process, in which end users rather than software engineers initiate model transformations, we do not find the transformation process too time-consuming: maybe the technological landscape has changed enough since 2001 for processing speed no longer to be quite such a concern.

Finally, it is interesting to view our forms-based approach from the perspective of domain-specific languages [18]: one might say that forms present a third style of DSL, in addition to—perhaps in between—the familiar textual and graphical styles.

8 Acknowledgements

We would like to acknowledge the other members of the CancerGrid team and related projects—James Brenton, Carlos Caldas, Marta Kwiatkowska, Peter Maccallum, Sylvia Nagl, Aadya Shukla, Matthew Snape, Tianyi Zang—for their contributions towards the ideas presented here. We are also very grateful to the Medical Research Council (grant number G0300648), Research Councils UK (grant number EP/F059345/1), the Engineering and Physical Sciences Research Council (grant number EP/H019944/1), the EU Framework Programme 7 (grant number 228436), and Microsoft Research for funding this work.

Some aspects of the CancerGrid project related to the work described here have also been published elsewhere. An early paper [2] set out the original vision, involving semantic web technology, computational grids, and computer-supported collaborative working; as discussed in this paper, many of these original ideas turned out to be unworkable in practice, and the project soon changed direction. A pair of related papers [5, 6] present formal specifications of an early prototype of our form-based approach, but programmed from scratch in Java rather than by configuring an off-the-shelf application (analogous to the first-principles implementation discussed in Section 4.2). A companion paper [12] focusses on the metadata aspects supporting shared semantics, rather than on the model-driven technology as described here. An extended abstract [13] briefly discusses the Kathmandu vaccinology study used as a running example in Section 4.

References

1. Blair, T.: Modernising government. UK Cabinet Office white paper CM 4310, UK Government (Mar 1999), http://www.archive.official-documents.co.uk/document/cm43/4310/4310.htm
2. Brenton, J., Caldas, C., Davies, J., Harris, S., Maccallum, P.: CancerGrid: Developing open standards for clinical cancer informatics. In: UK E-Science All Hands Meeting (2005)
3. Brenton, J., Davies, J., Gibbons, J., Harris, S.: Accelerating cancer research using semantics-driven technology. In: Microsoft eScience Workshop (Dec 2008)
4. Brooks, Jr, F.P.: The Mythical Man-Month. Addison-Wesley (1975)
5. Calinescu, R.: Model-based SOA generation for cancer clinical trials. In: Skar, L.A., Bjerkestrand, A.A. (eds.) OOPSLA Workshop on Service-Oriented Architectures. pp. 57–71. Portland, Oregon (2006)
6. Calinescu, R., Harris, S., Gibbons, J., Davies, J., Toujilov, I., Nagl, S.: Model-driven architecture for cancer research. In: Software Engineering and Formal Methods. pp. 59–68 (Sep 2007)
7. CancerGrid website. http://www.cancergrid.org/
8. CancerGrid metadata registry (cgMDR). http://www.cancergrid.org/index.php?option=com_content&id=8:mdrarticle
9. Cong, G., Fan, W., Jia, X., Ma, S.: PRATA: A system for XML publishing, integration and view maintenance. In: UK e-Science All Hands Meeting. pp. 432–435 (2006)
10. Crichton, C., Davies, J., Gibbons, J., Harris, S., Shukla, A.: Semantic frameworks for e-Government. In: Pardo, T., Janowski, T. (eds.) First International Conference on Theory and Practice of Electronic Governance (ICEGOV) 2007. pp. 30–39 (Dec 2007)
11. Crichton, C., Davies, J., Gibbons, J., Harris, S., Shukla, A., Tsui, A.: Semantics-driven development for electronic government applications. In: HICSS Workshop on Electronic Government (Jan 2009)
12. Crichton, C., Davies, J., Gibbons, J., Harris, S., Tsui, A., Brenton, J.: Metadata-driven software for clinical trials. In: ICSE Workshop on Software Engineering in Health Care. IEEE (May 2009)
13. Davies, J., Gibbons, J., Harris, S., Metz, J., Pollard, A.J., Snape, M.: Model-driven support for a vaccine study in Kathmandu. In: Microsoft eScience Workshop (Oct 2009)
14. Davies, J., Gibbons, J., Harris, S., Warzel, D.: Evolving health informatics: Semantic frameworks and metadata-driven architectures. In: Microsoft eScience Workshop (Dec 2008)
15. Davies, J., Harris, S., Crichton, C., Shukla, A., Gibbons, J.: Metadata standards for semantic interoperability in electronic government. In: International Conference on Theory and Practice of Electronic Governance (Dec 2008)
16. Earl, H.: Neo-tAnGo: A neoadjuvant study of sequential epirubicin + cyclophosphamide and paclitaxel ± gemcitabine in the treatment of high risk early breast cancer with molecular profiling, proteomics and candidate gene analysis. http://public.ukcrn.org.uk/search/StudyDetail.aspx?StudyID=1229 (2007), iSRCTN 78234870
17. Fegan, G.W., Lang, T.A.: Could an open-source clinical trial data-management system be what we have all been looking for? PLoS Medicine 5(3) (Mar 2008)
18. Fowler, M.: Domain Specific Languages. Addison Wesley (2010)

19. Harris, S., Calinescu, R.: CancerGrid clinical trials model 1.1. Tech. Rep. MRC/1.4.1.3, CancerGrid (2006), http://www.cancergrid.org/index.php?option=com_remository&func=fileinfo&id=185
20. ISO/IEC JTC1 SC32 WG2: ISO/IEC 11179, Information technology—metadata registries. http://metadata-standards.org/11179/
21. Kush, R.: Can the protocol be standardised? Tech. rep., Clinical Data Interchange Standards Consortium (2006)
22. McLaren, I., Wicks, T.: Developing generative frameworks using XML. In: Automated Software Engineering. pp. 368–372 (2001)
23. Mellor, S.J., Balcer, M.: Executable UML: A Foundation for Model-Driven Architecture. Addison-Wesley (2002)
24. Microsoft: InfoPath website. http://office.microsoft.com/en-us/infopath/
25. Microsoft: XML Schema Definition Tool (Xsd.exe) website. http://msdn.microsoft.com/en-us/library/x6c1kb0s(VS.80).aspx (2009)
26. Miller, J., Mukerji, J.: Model driven architecture: A technical perspective. Tech. Rep. ormsc/2001-07-01, Object Management Group (Jul 2001),
27. Moher, D., Schulz, K., Altman, D.G.: The CONSORT statement: Revised recommendations for improving the quality of reports of parallel-group randomised trials. The Lancet p. 357 (Apr 2001)
28. Poole, C., Earl, H.: NEAT: National breast cancer study of epirubicin plus CMF versus classical CMF adjuvant therapy. http://public.ukcrn.org.uk/search/StudyDetail.aspx?StudyID=643 (2001), iSRCTN 42625759
29. Poole, C., Howard, H., Dunn, J.: tAnGo: A phase III randomized trial of gemcitabine in paclitaxel-containing, epirubicin-based adjuvant chemotherapy for women with early stage breast cancer. http://public.ukcrn.org.uk/search/StudyDetail.aspx?StudyID=661 (2004), iSRCTN 51146252
30. Sierra, J.L., Fernández-Valmayor, A., Fernández-Manjón, B.: A document-oriented paradigm for the construction of content-intensive applications. Computer Journal 49(5), 562–584 (2006)
31. Steele, G.L.: Objects have not failed (Nov 2002), position statement for panel at OOPSLA,
32. Stell, A., Sinnott, R., Ajayi, O.: Supporting the clinical trial recruitment process through the grid. In: UK e-Science All Hands Meeting. pp. 61–68 (2006)
33. Sullivan, L.: The tall office building artistically considered. Lippincott's Magazine (Mar 1896)
34. University of Birmingham School of Medicine: How Birmingham researchers are taking a measured look at medical treatments. Medlines 4 (Jul 1997), http://www.publications.bham.ac.uk/medlines/1997b/a_medpg3.htm
35. US National Cancer Institute: caCORE software development kit. http://ncicb.nci.nih.gov/infrastructure/cacoresdk (2006)
36. US National Cancer Institute: Cancer Biomedical Informatics Grid. https://cabig.nci.nih.gov/ (2006)
37. Wong, P.Y.H., Gibbons, J.: A process-algebraic approach to workflow specification and refinement. In: Software Composition (2007)
38. Wong, P.Y.H., Gibbons, J.: On specifying and visualising long-running empirical studies. In: International Conference on Model Transformations (ICMT). LNCS, vol. 5063, pp. 76–90. Springer-Verlag (Jul 2008)
39. Wong, P.Y.H., Gibbons, J.: Property specifications for workflow modelling. Science of Computer Programming (2010)
40. Wong, P.Y.H., Gibbons, J.: Formalisations and applications of BPMN. Science of Computer Programming 76, 633–650 (2011)