CancerGrid: Metadata-Based Model-Driven Engineering for Clinical Trials

Can you give us a brief high-level summary of the project?

Randomized controlled trials are the gold standard for experiments in medicine. Two or more treatments are being compared, often a new treatment against the best existing treatment; subjects are recruited for the trial, and stratified into groups based on criteria such as age, gender, and lifestyle; patients in each group are allocated one or other treatment at random; and the results analysed to determine whether there are any statistically significant effects.

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 analysed. Two particular issues that the CancerGrid project addressed involve data integration and tool generation.

Data integration is required because medical researchers need to be able to combine the results from multiple independently conducted trials, a process known as meta-analysis. Maybe a single trial shows a small effect, but it is not statistically significant; however, when put together with the results from other sufficiently similar trials, there is enough data to demonstrate significance. Supporting meta-analysis entails recording and propagating metadata describing the "semantics" of the data. For example, it is not sufficient to record that blood pressure is 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 when. 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 generation, current standard practice in clinical trials management 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, all familiar to advocates of MDE, which I discuss next.

Can you summarize what it was about MDE that helped you make a difference in this case?

Here are four problems caused by the current standard practice of manual development or configuration of software to support a trial.

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 often too busy to respond immediately, and it is not uncommon for the trial to have to start on paper because the software is not ready.

Secondly, it is costly. This is not such a problem for a big "phase III" trial by a pharmaceutical company pursuing regulatory approval: the study will have thousands of participants and a stable design, so the software development will form only a small proportion of the overall cost, and is likely to be recouped in sales over the lifetime of the drug. However, it is a problem for early-phase exploratory studies and late-phase post-approval studies: the former are much smaller, more dynamic and inherently risky, as animal models are an unreliable predictor of efficacy in humans; the latter are typically funded by charities, governments and NGOs in academic settings on a tight budget. Even then, many promising drugs are not brought to market because the return on the drug outweighs the cost of approval.

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 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 of the trial, rather than merely a textual description – then both the prose and the code could be 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, 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 approach!

Where there any challenges along the way? How did you handle them?

We did encounter a number of non-technical obstacles to the MDE approach we espoused. For example, we had little traction in supporting the dynamic aspects of trial execution by exploiting a workflow model in the trial protocol. This is partly because it is not a particular pain point for the communities we are working with: current practice in randomized allocation of patients to treatment arms is for a statistician metaphorically to produce in advance a sequence of decisions in envelopes, which is a sufficiently low-technology practice to work well even in the most rudimentary of clinical contexts. Similarly, trials units generally already have well-established processes for exporting clinical data into statistical packages for subsequent analysis, and there was no obvious gain from integrating this aspect with the rest of the model-driven chain.

We also met objections due to social and political interests, especially an investment – whether individual or institutional – in more manual development techniques. The leader of a trials unit who has to find funding for database team may welcome automating away their job; but if the overworked trials unit leader delegates to the database team leader the discussion about whether MDE is a good thing, automation may look much less appealing.

What has happened since the project was completed? What are the next stages for your project?

The original CancerGrid project was funded by the UK MRC for three years, 2005 to 2008, with my colleague Jim Davies at the University of Oxford leading the software engineering aspects of the work. Subsequent collaborative projects applying the approach include: Accelerating Cancer

Research Using Semantics-Driven Technology, funded by Microsoft Research, exploring the extension from phase III to early-phase studies; Evolving Health Informatics, funded by RCUK, working with vaccinology colleagues to demonstrate applicability to infectious disease control; Hospital of the Future, funded by the UK EPSRC, aiming to improve patient outcomes through information-driven management; the Data Support Service, funded by the UK MRC, to retrospectively catalogue the data collected in some of their valuable long-running studies; and the Union of Light-Ion Centres in Europe, funded by EU FP7, to curate experimental results in particle therapy. In addition, we have been collaborating with local colleagues on a pro bono basis.

The project title is somewhat of a misnomer. As the name suggests, it was born during a flurry of enthusiasm in the UK for “grid computing” – large-scale distributed computing and storage clusters, especially for scientific applications such as climate modelling and astronomy. These technologies turned out to be impractically immature for us at the time; moreover, the large-scale aspects were not relevant. And although the original proposal was written with oncologist colleagues at Cambridge, nothing from the SE perspective is specific to cancer – as some of the subsequent projects attest. We did own and run the cancergrid.org website for some years, but I see that it has since been usurped by an unrelated organization.

In fact, we were 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 – interact. 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. We explored some preliminary ideas in this area, and had some discussions with the UK Public Sector Object Model group and with the Scottish Government, but I believe that these stalled due to lack of capacity on our side.

Would you use MDE for similar projects again?

Yes, absolutely! I personally got distracted by other activities, and I have not really been involved in the endeavour for some years; but Jim Davies and his group are very definitely continuing this work, continuing to use model-driven generation and especially continuing to explore the collection, curation, and exploitation of semantic metadata.

This article is based on the paper "The CancerGrid Experience: Metadata-Based Model-Driven Engineering for Clinical Trials" by Jim Davies, Jeremy Gibbons, Steve Harris, and Charles Crichton (Science of Computer Programming 89B:126-143, 2014, DOI 10.1016/j.scico.2013.02.010).

Jeremy Gibbons is Professor of Computing at the University of Oxford, where he leads the Algebra of Programming research group, and teaches on the part-time professional postgraduate master’s programme in Software Engineering. His research interests are in programming languages, especially functional programming and patterns in programming. He is Editor-in-Chief of the Journal of Functional Programming, and a member of IFIP Working Group 2.1 on Algorithmic Languages and Calculi, and of IFIP Working Group 2.11 on Program Generation.