Formal Methods for Future Interoperability

Jim Davies and Jeremy Gibbons

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK.

{jim,davies,jeremy.gibbons}@comlab.ox.ac.uk

Abstract: Interoperability is a key challenge in software engineering, whether expressed in terms of the compatibility of different systems and protocols, in terms of compliance to industry standards, or—increasingly—in terms of the ability to share and re-use data gathered in different contexts. Formal methods are mathematical techniques for the precise description of systems properties and behaviour, and have an important role to play in the future provision of interoperable systems and data. This paper describes that role, and examines the implications for present-day training and education.

Categories and subject descriptors: D.2.4 [Software engineering]: Software/program verification – Formal methods; K.3.2 [Computers and education]: Computer and information science education – computer science education; J.1 [Administrative data processing]: Government

General terms: Standardization, design, verification

Keywords: Metadata, model-driven development, semantic frameworks

INTRODUCTION

Interoperability is a fundamental concern for users and developers of information technology. The ability to create a document on one computer, and open it on another, is now seen as a basic requirement. Likewise the ability to create an appointment, send it to another person's calendar, and have it appear with the correct date, time, and list of participants. And over a period of years, we have seen the industry arrive at a set of common standards that allow this to happen—most of the time.

Perhaps the most important of these are the standards that define the Internet itself, standards like TCP/IP. It is now difficult to imagine or remember a time when protocol stacks from different manufacturers would behave together unpredictably. These days, at the lower levels, in terms of the basic protocols, the Internet just works. And the same is true of our mobile phones, our smartcards, and even our electronic voting systems—most of the time.

But interoperability at the lowest level is not enough. As the systems we build become ever closer to our lives, as we become increasingly reliant upon them for our work, our personal security, and our understanding of the world, we start to need interoperability at higher and higher levels of abstraction and meaning. And we need to be able to depend upon this interoperability—all of the time.

When we pay our electricity bill at our bank, we want to know that our account at the electricity provider will be credited. When we arrive at an immigration desk, we want to know that the information that the officer has is accurate, and that we will not be mistaken for a dangerous criminal. When we visit a hospital, we want the correct treatment for our condition, not a prescription—or even an operation—intended for someone else.

Not so long ago, a computer's version of past or future events was easily corrected or challenged: it was widely accepted that "computers make mistakes", and mistakes were generally easy to detect and rectify. A customer would have paper copies of electricity bills and bank statements, and any discrepancy might be corrected with a visit to the local branch, or a phone conversation with a human operator.

If the information at the immigration desk, or on the hospital system, was incorrect, then we might be inconvenienced, but our hope would be that nothing too serious would happen without additional checks, a paper trail, and a review of physical evidence. But in this millennium, the additional checks will be made using computers, there will be no paper trail—no paper bills, statements, tickets, prescriptions, or visas—and as far as any evidence is concerned, the electronic version will be definitive; it will be the master copy.

This may seem a frightening, even paranoid, vision—a world in which any normal, law-abiding citizen could be plunged into a Kafkaesque nightmare of persecution without warning or hope of redress—but it is also a call to arms: this computerisation of our lives will happen, it is happening, and it many respects it has already happened; the challenge that we face is to ensure that the behaviour of our computing systems is correct.

And this is a task for formal methods: it is time for formal methods to (help) save the world. But are formal methods, as we know them, up to the job? Are we teaching the right techniques? Is our research sufficiently advanced? Is what we know, and what we do, properly integrated into the processes of systems development, operation, and maintenance?

The answer, regrettably, is no. And much of the problem lies with interoperability at higher levels of abstraction: by this we mean not questions of programming language semantics—checks that variables will achieve or maintain certain values—but questions of what those variables and values mean outside of the current context:

what they mean to other systems, and what they mean to stakeholders, owners, developers, and users.

In this paper, we explore the nature of this problem, and formulate a grand challenge for formal methods research and education, one that we must address if we expect to live in a world where computing systems "just work", in the ways that we want and believe them to: a world in which systems are sufficiently correct, accurate, and accountable to be trusted with information that can determine the course of our lives.

ELECTRONIC GOVERNMENT

One domain in which interoperability is of obvious, overriding importance is *electronic government*— the use of information technology "to improve outreach, access of information, reduced corruption, increased transparency, greater convenience, revenue growth and cost reductions" [13], "combined with organisational change and new skills in order to improve public services and democratic processes" [6], and "to promote more efficient and effective government, facilitate more accessible government services, encourage interaction between users of public services and providers" [12].

Systems development in electronic government involves more than a literal translation of existing services and processes into electronic form. The familiar problems of large-scale projects are exacerbated by three main factors: the likelihood of conflict and misunderstanding between different stakeholder groups; the fact that requirements are linked to changes in policy and legislation; and the expectation that data and processes should be accessible, and immediately interoperable with those in other initiatives.

An example commonly quoted in the electronic government community in the UK involves the automatic provision of free school meals for the children of those in receipt of specific social security benefits, which remains a manual operation. It is a failure of interoperability that the task of obtaining the relevant information from one department, and translating it into terms understood by another, in order to prove eligibility, is something that has to be undertaken manually by the citizen concerned.

Electronic government carries expectations of transformation, often in connection with hopes for a better society. The use of words such as 'transformation', 'access', 'transparency', 'change', 'democracy', and 'interaction', suggest specific domain challenges, with significant implications for the processes of software design and development.

In particular, electronic government requires a high degree of formalisation and computerisation of semantics—again, not the mathematical semantics of programming languages, but the intended interpretation and usage of system artefacts: variables, values, objects, functions, services, models, forms, and documents. Such a semantics can be expressed by communicating, in a structured fashion, the role of artefacts in giving each other meaning.

For example, the meaning of a particular item of data, stored in a database, might be explained partly by association with the question on the form used to collect that data, and partly by a document describing the context in which it was collected. The form and the document may themselves be associated with other data, or other artefacts describing provenance, versioning, ownership, purpose, and intended use.

An informal, or implicit treatment of this kind of semantics is sufficient only when the concepts are straightforward, the user community is small or homogeneous, and the period of time over which shared understanding must be maintained is short. For systems of any complexity, communities of any size, or initiatives that are intended to last for many years—for the kind of systems developed for electronic government—a more formal, explicit approach is required.

The lack of explicit, managed semantics is a contributing factor in the continuing failure of many electronic government initiatives. Most people can quote at least one high-profile disaster, in which a large, public sector project has failed to deliver. (Indeed, on the very day of writing this paragraph, the UK government announced the cancellation of a half-complete project to digitise and make accessible 171 years' worth of genealogical records [7].)

The mere existence of explicit, managed semantics is not enough, however: we need also to ensure that it is properly reflected in the behaviour of the systems that we build and use. If it is to mean anything in practice, then the semantics must be faithfully incorporated at every stage of systems development, operation, and maintenance.

SEMANTIC FRAMEWORKS

We believe that a big step towards addressing the interoperability challenges of electronic government can be made by integrating ideas from *data semantics* and *model-driven development*, an integration we call a *semantic framework*. Moreover, we claim that semantic frameworks both provide an interesting new domain for, and can derive great benefit from, work in formal methods.

Metadata-based

Robust, trustworthy, and transparent information systems require the careful consideration and representation of the semantics of the information they record; a structured, computable representation is essential if we wish to adopt and maintain rich terminologies across multiple initiatives.

Conflicts and misunderstandings about the semantics of data can be resolved, or at least identified at an earlier stage, if aspects of structure, functionality, and interpretation are conveyed through the use of models. This is standard practice in software engineering; however, the

audience for the model is usually quite restricted, and thus much of the detail, or semantic metadata, may be left implicit.

For electronic government, in particular, it is a requirement that models may be validated, so that public servants can be held accountable; it is therefore essential that the models are both comprehensive and comprehensible—the metadata should be properly recorded, and accessible to the stakeholders and users, as well as to the developers. Furthermore, the connection between data and metadata must be maintained at every stage of acquisition and processing.

Model-driven

The dynamically evolving context of policy and legislation, the greater requirements for accountability and transparency, and the sheer scale of many electronic government initiatives, all encourage the automatic generation of system artefacts—such as forms, services, queries, database schemas—or even complete system implementations, from abstract models.

Information systems are modelled as matter of course, but often only informally, using fragments of specification, written in natural language, and presented as reports, spreadsheets, and diagrams. These are partial descriptions, often containing apparent contradictions, and there is no prospect of using these to generate a system automatically. Yet these are the documents that inform decisions such as those on whether to proceed, on project scope, on supplier selection, and on contract fulfilment, and it is here that a semantic framework can start to produce real benefit.

In development, more formal models—typically, object models and service descriptions—can present precise descriptions of structure and functionality in which data attributes have an accessible, computable semantics, and terms have an agreed meaning. It may then be possible to determine programmatically—at the design stage, or after deployment—whether two systems are holding data that has exactly the same semantics.

One way to represent the semantic information required, and to facilitate programmatic access, is to represent the various aspects of semantics using models of usage. We can identify three particularly useful kinds of model: *ontologies*, models which explain the meaning of a metadata item in terms of named relationships to other elements; *applications*, models in which the item appears in context: for example, in the context of a design document, or a form template; and *transformations*, models which explain how data collected against one set of elements can be transformed to fit another. Although only the first of these is usually seen as defining or recording meaning, the others also have semantic import: meanings are sometimes best expressed, and will evolve, through usage.

If we can develop and configure systems through automatic transformations, then we have a means of automatically translating agreements on intended meaning—developed iteratively, in collaboration with stakeholders—into the behaviour of the working system. This would work to increase trust, to promote standards adoption, and—by reducing the cost of developing alternative or additional functionality—to facilitate the development of systems that meet the varying needs of a wide range of stakeholders.

Semantic frameworks

The ideas of metadata-based and model-driven development together make what we call a *semantic framework*. A practical semantic framework can be defined in terms of constructs at three different levels: *terminology services*, *metadata registries*, and *model repositories*. The first level presents a collection of defined terms, structured in a way that suits one or more possible applications. For example, a terminology for education might include terms such as ‘institution’ and ‘qualification’, record that the terms ‘university’ and ‘high school’ denote particular kinds of institution, and record also that the terms ‘master’s degree’ and ‘international baccalaureate’ are related in some way to the notion of institution.

The second level presents a collection of metadata elements, each of which describes a measurement or observation. A metadata registry for education might include elements such as institution attended, full title of degree awarded, and result obtained. Each element may be related to one or more terms in the underlying terminology, and additional semantic information is provided by informal explanations of intended purpose and an association with a domain of possible values. The registry also records relationships between elements, such as equivalence, specialisation, and versioning.

The third level presents re-usable models for the definition of information artefacts, such as database schemas, service descriptions, forms, queries, and reports. A model repository for education might include models of admissions forms, study transcripts, and spreadsheets for reporting registration and progress data to national agencies. The fields on the forms, the entries on the transcripts, and the columns on the spreadsheets may be described, and given computable semantics, by linking them to the metadata elements in a metadata registry.

The State of the Art

Metadata standards are already starting to appear in the area of electronic government. An international standard for metadata registries, ISO/IEC 11179 [9], is in its third version, and has been adopted by—amongst others—the Australian Institute of Health and Welfare [2], the Canadian Institute for Health Information [3], and the US Departments of Homeland Security and Justice [11]. The Scottish Government is now able to use a standard collection of metadata to drive procurement—prospective

suppliers of information systems can be required to conform to agreed, customer-supplied terminology and data semantics [10]. There is as yet, however, little progress in terms of model-driven development for electronic government applications.

In the domain of healthcare, more progress has been made. Specifically, in translational cancer research—in which scientific advances depend upon the effective combination of clinical and research information from a variety of sources—metadata registries and model-driven techniques are being used together to guarantee the interoperability of data collected in different contexts. The US National Cancer Institute and the UK CancerGrid Project [4] have successfully demonstrated semantic interoperability and automatic data integration across national and institutional boundaries. The semantic framework approach has since been adopted by leading players in the pharmaceutical industry.

EDUCATION AND TRAINING

The increasing importance of interoperability, and compatibility with the needs and expectations of users and other stakeholders, and the consequential shift towards metadata-based, model-driven approaches, has significant implications for the teaching of software engineering, and for the teaching of formal methods in particular.

Software engineers will need expertise in generative programming techniques—writing software to write software, rather than programming directly at the level of a fixed implementation. They will need experience in precise modelling at higher levels of abstraction. Most importantly, they will need the ability to create and maintain models—and generators—through effective, continuing collaboration with domain experts.

Not only will precise modelling become an essential component of the software engineering toolkit, it will also be important to anyone who needs to understand or validate the documents and artefacts that determine the meaning and behaviour of software systems—experts, managers, customers, auditors, and even ordinary citizens.

Both professional software engineers and these other, lay, stakeholders will need education and training in formal methods. In neither case, however, will this be the kind of education and training currently being delivered, or envisaged in curriculum proposals such as SWEBOK [8] and the GSwERC [1].

Formal methods have traditionally been seen as most applicable in limited domains: typically high-integrity, safety-critical, embedded systems. The emphasis has been either upon the painstaking, manual derivation of fixed implementations from immutable specifications, or upon the post-hoc verification of program code or hardware designs, and these are less relevant when our systems are to be generated, automatically, from high level models.

Instead, all of those directly involved in systems development will need education and training in precise, abstract modelling, in conceptual modelling, and in the analysis of the resulting artefacts—at least to the level of being able to understand and act upon feedback from analysis tools.

In addition, professional software engineers will need education and training in translation and generation technology, in higher-order, declarative techniques, in semantic frameworks, and in requirements engineering techniques that take advantage of the immediate connection between abstract models and system behaviour.

ACKNOWLEDGEMENTS

This position paper draws on an earlier paper [5], with additional contributions from Aadya Shukla and Steve Harris.

BIBLIOGRAPHY

1. Applied Systems Thinking Institute. *Graduate Software Engineering Reference Curriculum*. 2009. http://www.asysti.org/issechome.aspx
2. Australian Institute of Health and Welfare. *Metadata Online Registry (METeOR)*. http://meteor.aihw.gov.au/
3. Canadian Institute for Health Information. *CIHI Data Dictionary*. http://secure.cihi.ca/ddexternal/
4. CancerGrid Project. http://www.cancergrid.org/
5. Charles Crichton, Jim Davies, Jeremy Gibbons, Steve Harris, and Aadya Shukla. *Semantics Frameworks for e-Government*. In *International Conference on e-Government*. December 2007.
6. Commission of the European Communities. *The Role of eGovernment for Europe's Future*. September 2003. http://ec.europa.eu/information_society/eeurope/2005/doc/all_about/egov_communication_en.pdf
7. David Hencke and Robert Booth. *Ancestry Hunters Stuck in Past as Web Project Fails*. *The Guardian*, 16th August 2008. http://www.guardian.co.uk/technology/2008/aug/16/genealogy.records
8. IEEE *Guide to the Software Engineering Body of Knowledge*. 2007. http://www.swebok.org/
9. ISO/IEC 11179. *Information Technology – Metadata Registries (MDR)*. http://metadata-stds.org/11179/
10. Scottish Government. *OSIAF: Openscotland Information Age Framework*. August 2006. http://www.scotland.gov.uk/Publications/2006/08/24092730/
11. US Departments of Homeland Security and Justice. *National Information Exchange Model (NIEM)*. http://www.niem.gov/
12. Pacific Council on International Policy. *Roadmap for E-Government in the Developing World*. April 2002. http://www.pacificcouncil.org/pdfs/e-gov.paper.f.pdf
13. World Bank. *E-Government Practice*. http://go.worldbank.org/C81XI6R6F0