Transatlantic TUmour MOdel Repositories

D4.2.1

State of the art review on mark-up languages and model wrap-up directions tailored to TUMOR models

Project Number: FP7--IST-247754

Deliverable id: D4.2.1

Deliverable name: State of the art review on mark-up languages and model wrap-up directions tailored to TUMOR models

Submission Date: 31 March 2011

Information Society Technologies

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	TUMOR
Project Full Name:	Transatlantic TUmour MOdel Repositories
Document id:	D4.2.1
Document name:	State of the art review on mark-up languages and model wrap-up directions tailored to TUMOR models
Document type (PU, INT, RE)	PU
Version:	1.0
Submission date:	31 March 2011
Editor: Organisation: Email:	David Johnson University of Oxford david.johnson@comlab.ox.ac.uk

Document type PU = public, INT = internal, RE = restricted

ABSTRACT: This deliverable document reviews the state-of-the-art markup languages that can be used to wrap up and annotate cancer models that will be stored and made available in the TUMOR digital model repository. The current state-of-the-art biological modelling languages are reviewed, and we discuss both their potential use for cancer modelling and their pitfalls when applied to the cancer modelling domain. Next, we review a range of XML-based metadata vocabularies that could be integrated into a new markup language for cancer modelling. A brief review of relevant medical ontologies follows, and we conclude with a high-level design for *TumorML*, a new markup language for cancer modelling that we will develop within the TUMOR project to address the pitfalls of the current state-of-the-art.

KEYWORD LIST: design directions, state-of-the-art review, markup, XML, cancer modelling

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	9 March 2011	Draft	David Johnson
0.2	20 March 2011	Draft	David Johnson
0.3	22 March 2011	Draft	David Johnson
0.4	24 March 2011	Draft	David Johnson
0.5	24 March 2011	Draft	Jonathan Cooper
0.6	24 March 2011	Draft	David Johnson
0.7	25 March 2011	Draft	David Johnson
0.8	28 March 2011	Draft	Stelios Sfakianakis
0.9	30 March 2011	Pre-final	David Johnson
1.0	31 March 2011	Final	David Johnson

List of Contributors

- All consortium
- David Johnson (UOXF.BL)
- Jonathan Cooper (UOXF.BL)
- Steve McKeever (UOXF.BL)
- Stelios Sfakianakis (FORTH)
- Vangelis Sakkalis (FORTH)
- Kostas Marias (FORTH)
- Giorgos Manikis (FORTH)
- Ioanna Lykourentzou (ICCS)
- Dimitra Dionysiou (ICCS)
- Georgios Stamatakos (ICCS)
- Thomas Deisboeck (MGH-CViT)

Contents

1 EXECUTIVE SUMMARY 5
2 INTRODUCTION 6
3 CURRENT STATE-OF-THE-ART MARKUP LANGUAGES FOR BIOLOGICAL MODELLING 8
THE SYSTEMS BIOLOGY MARKUP LANGUAGE 8
CELLML 9
FIELDML 11
*INSILICO*ML 11
DISCUSSION 12
4 EXISTING REUSABLE MARKUP VOCABULARIES 15
MATHML 15
JOB SUBMISSION DESCRIPTION LANGUAGE 16
DUBLIN CORE 19
RDF VCARD 20
FOAF 22
SIMULATION EXPERIMENT DESCRIPTION MARKUP LANGUAGE 24
HL7 CLINICAL DOCUMENT ARCHITECTURE 24
MEDICAL MARKUP LANGUAGE 26
XML SIMPLE CONCEPTUAL UNIFIED FLOW LANGUAGE 27
BUSINESS PROCESS EXECUTION LANGUAGE 28
5 ONTOLOGIES FOR THE CLINICAL MEDICINE AND CANCER DOMAIN 31
NATIONAL CANCER INSTITUTE THESAURUS 31
INTERNATIONAL CLASSIFICATION OF DISEASES (ICD) 32
SYSTEMATIZED NOMENCLATURE OF MEDICINE – CLINICAL TERMS 33
ACGT MASTER ONTOLOGY 34
6 TUMORML: A MARKUP LANGUAGE FOR COMPUTATIONAL CANCER MODELLING 36
CONCEPTUAL DESIGN 36
MARKUP FOR CURATING CANCER MODELS 38
MARKUP FOR INTERFACING WITH MODELS 39
MARKUP FOR CONNECTING MODEL INTERFACES 40
DELIVERABLE XML SCHEMAS, STYLESHEETS, AND SUPPORT TOOLS 41
7 CONCLUSION 42

1 Executive Summary

The TUMOR project aims at developing a European *clinically oriented* semantic-layered cancer digital model repository from existing EC projects that will be interoperable with the US grid-enabled semantic-layered digital model repository platform at CViT.org (Center for the Development of a Virtual Tumor, Massachusetts General Hospital (MGH), Boston, USA) which is NIH/NCI-caBIG compatible. This interoperable, CViT interfaced, environment will offer a range of services to international cancer modellers, bio-researchers and eventually clinicians aimed at supporting both basic cancer quantitative research and individualized optimization of cancer treatment. This 'Transatlantic' project will therefore be the starting point for an international validation environment that will support joint applications, verification and validation of the clinical relevance of cancer models.

The purpose of this deliverable is to review the state-of-the-art markup languages that can be used to wrap up and annotate cancer models that will be stored and made available in the TUMOR digital model repository. The current state-of-the-art biological modelling languages are reviewed; we discuss their potential use for cancer modelling and their pitfalls when applied to the cancer-modelling domain. Next, we review a range of XML-based metadata vocabularies that could be integrated into a new markup language for cancer modelling. A brief review of relevant medical ontologies follows, and we conclude with a high-level design for *TumorML*, a new markup language for cancer modelling that we will develop within the TUMOR project to address the pitfalls of the current state-of-the-art.

The development of *TumorML* will contribute to enabling some of the key aims within the TUMOR project. Firstly, by annotating cancer models with appropriate document metadata, digital curation will be facilitated that will make publishing, search and retrieval of cancer models easier for researchers and clinicians using the TUMOR digital repository developed under WP3. Second, markup will be used to describe abstract interfaces to published implementations allowing execution frameworks to run simulations using published models, as required by WP5. Finally, *TumorML* markup will facilitate the composition of compound models, regardless of scale and source, enabling multiscale models to be developed in a modular fashion, and models from the US CViT DMR to be integrated with EC models in the transatlantic scenarios developed in WP2.

2 Introduction

The TUMOR project aims to develop a European clinically oriented semantic-layered digital model repository for cancer models. The repository will store models provided by other EC projects, namely the Advancing Clinico Genomic Trials on Cancer (ACGT) and the Clinically Oriented Translational Cancer Multilevel Modeling (ContraCancrum) projects. One of the major aims of the TUMOR project is to design the digital repository to interoperate with the US counterpart repository developed by the Center for the Development of a Virtual Tumor (CViT) project led by the Massachusetts General Hospital (MGH) in Boston, USA.

WP4 focuses two-fold on developing interoperable interfaces between the two repositories. Firstly, the TUMOR project aims to design the EC repository to be able to functionally operate seamlessly with the CViT Digital Model Repository (DMR). This will be achieved by the development of a set of Web services to expose the EC repository's functionality to CViT DMR users, and by using the CViT DMR's existing exposed Web service functionality. Secondly, and the focus of this deliverable, the TUMOR project will develop a simulation markup language specifically targeted at the cancer modelling domain. The overarching goal is to provide cancer researchers on both sides of the Atlantic (through their respective research platforms) more open access to computational cancer models shared amongst the international modelling community.

The development of a markup language for cancer models will enable the provision of two features:

1. By providing an expressive metadata vocabulary researchers will be able to appropriately curate their models and publish them to an audience of research peers and clinicians wishing to trial published models
2. Markup will be developed to describe abstract interfaces to the computational execution of the models. These abstractions will be mapped to the appropriate biological entities that could be used to couple cancer models together.

To demonstrate the fulfilment of (1), models taken from ACGT and ContraCancrum will be published directly to the EU repository by wrapping the computational components (as source code and/or executable binaries) in the newly developed markup language. Through the Web services delivered by D4.1.2, CViT DMR models will be imported into the EC repository where the US model metadata will be appropriately translated for storage by TUMOR.

To demonstrate the fulfilment of (2), WP5 (Integrated, interoperable workflow environment) will, through automatic interpretation of the model markup, provide functionality to couple models using a graphical workflow tool. The tool will allow execution of the aggregate model as a workflow. As an exemplar for the 'transatlantic' aim of the project, a model taken from the US repository will be coupled with one provided by ACGT or ContraCancrum.

Digital repositories for computational models are not novel, as demonstrated by a number of model repositories including E-Cell, the CellML and FieldML repositories, BioModels and the CViT DMR. However semantic integration for specific domains is still required where most of the aforementioned repositories targets select research areas. The TUMOR project will provide a repository for European-based and international researchers for cancer models.

In computational biology, there is diverse range of programming and descriptive languages that span across different biological domains and scales. This creates challenges for model reuse and composition, since each model implementation, even if available, may use a completely different technological framework. Combining models may therefore require porting models to a new framework, or re-implementing them, both costly and error prone activities. Before the year 2000, there were no unified efforts towards standardized

languages for describing models. Markup languages for computational biology emerged soon after the turn of the millennium with the SBML (Systems Biology Markup Language) and CellML research programmes.

Generally speaking, all application-specific markup languages are based on the eXtensible Markup Language (XML). XML emerged as a popular choice for computer-based language definition in the late 1990's as it defines a standard syntax on to which other vocabularies can be built. This allows language-specific parsers to reuse the standard XML parsing routines for processing XML documents. XML-based languages are therefore well suited as software-neutral information exchange formats. XML can be thought of as the base alphabet and grammar of a language. What raw XML lacks is semantics or definitions to provide context and application.

This is provided in part by specific vocabularies built on XML, which define element and attribute names and the structural relationships between them. Multiple XML vocabularies can be combined within a single document, enabling the development of various languages targeted at specific narrow domains of discourse that can be incorporated into a compound language.

Further semantic metadata can be added to XML documents through the use of the Resource Description Framework (RDF). This standard from the World Wide Web Consortium enables descriptions (i.e. metadata) to be associated with any resource, such as a whole model, or a specific element within a model. Similarly to XML, RDF does not define specific metadata items, but rather provides a standard framework onto which metadata vocabularies can be hung. Some benefits of having this standard framework are:

- **It provides a common *`attribute=value`* data model for the metadata**. All metadata expressed in RDF can be presented as a series of attributes (i.e. properties of the resource) and their values.
- **It therefore provides an extensible method for storing metadata of increasing complexity**. Some metadata properties will have simple values. Other metadata properties will have complex values. In the latter case, the value of the metadata property is itself considered a resource, and additional metadata properties are stored about it. Furthermore, the system is open in that additional properties may be defined and incorporated later – the whole vocabulary does not need to be defined up-front.
- **This openness makes it possible for applications that don't understand the whole model to process the metadata**. There exist tools that understand general RDF and can parse it to build databases, knowledge stores, and the like, making inferences from the semantic information content. Where they understand a specific metadata vocabulary, more complex processing is possible, but even for other metadata items the *`attribute=value`* model still allows logical reasoning about relationships between resources.

3 Current State-of-the-art Markup Languages for Biological Modelling

The need for fusing models together has been outlined in D2.1 (Specification and design of clinically oriented transatlantic scenarios), but how can we facilitate connecting disparate models together? Markup languages for modelling biological systems (based on XML) emerged in the early 2000s to address the problems associated with no single emerging standard for describing biological models. Four major markup languages have gained prominence in recent years each of which aims to tackle the problems associated with interoperability of biological models in different ways. The markup languages reviewed here are the Systems Biology Markup Language, CellML, FieldML and *insilico*ML.

The Systems Biology Markup Language

Maintained by: SBML Community

Latest version: Level 3 Version 1 Core

Availability: http://sbml.org/Documents/Specifications#SBML_Level_3

Description

The Systems Biology Markup Language (Hucka et al., 2004), commonly referred to as SBML, is a domain-specific XML markup language that addresses biochemical processes at the molecular scale. The motivations for SBML were three-fold:

1. Multiple tools are usually used to develop models in systems biology. This is due to the fact that different modelling tools lend different advantages. For example, one tool may provide a useful graphical modelling interface, whilst another may implement facilities for finer grained modelling. Where a single software package does not provide the strengths afforded by both, there is the need for a common file format for both tools to be able to read and write
2. Electronic versions of models often accompany written publications in peer-reviewed journals. Due to the diverse numbers of modelling software packages used to develop models, it was commonly found that researchers wishing to test models and re-run experiments either had to use the original modelling software, or to re-encode models to their preferred platform
3. Models encoded in multiple languages tend to out live the modelling software tools in which they are implemented. No single modelling language was present to enable future interoperation with systems biology models.

The aforementioned aims are quite generalised. However, the original authors explain that SBML does not aim to be a generic modelling language to cover quantitative models. They recognise that the common understanding of biological processes evolves quickly and as such suggest that a modelling language for systems biology be domain-specific and structured to represent the consensus of current understanding in the field. This aims to enable the state-of-the-art modelling tools in systems biology to use a common language in which to communicate models, rather than having a generalised modelling language for biological and/or computational modelling.

SBML is developed as a set of incremental levels, where each subsequent level supersedes the former. This was to allow the language to be adopted quickly, and to evolve with the requirements of the representation and understanding of systems biology. Here we briefly describe the components of the Level 3 specification of SBML that is described in full in Hucka et al (2010).

SBML is comprised of a collection of optional components. A model definition lists a set of mathematical functions, unit definitions, compartments (a container that may or may not represent a physical structure), species (a set of entities within a compartment, such as chemical substances, that participate in some kind of reaction), global and local parameters, initial assignments that define the initial conditions of the model, rules that constrain the models behaviour, constraints, reactions (statements that describe some kind of process such as biochemical transformations or transport), and events to capture instantaneous changes within the model. To describe the mathematical components in SBML, the language utilises Content MathML (Carlisle et al, 2009), an XML language for describing mathematical formulae. While MathML supports encoding of arbitrary mathematical formulae, the SBML specification restricts the MathML vocabulary that may be used within SBML models to facilitate implementation. Typically the mathematics used to model systems biology is in the form of declarative formulae such as ordinary differential equations (ODEs) and differential-algebraic equations (DAEs), and the markup used can adequately describe such equations. Spatially varying models using partial differential equations (PDEs), however, are not supported. SBML also provides facilities to associate metadata with models in order to properly curate models within online databases.

The community of SBML users and developers have worked extensively to create a wide range of tools and infrastructure to support SBML models.

References

Hucka, M., Finney, A., Bornstein, B.J., Keating, S.M., Shapiro, B.E., Matthews, J., Kovitz, B.L., Schilstra, M.J., Funahashi, A., Doyle, J.C., Kitano, H., 2004. Evolving a Lingua Franca and Associated Software In- frastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project. Systems Biology 1.

Hucka, M., Hucka, M., Bergmann, F., Hoops, S., Keating, S., Sahle, S., Wilkinson, D., Hucka, M., Bergmann, F., Hoops, S., Keating, S.M., Sahle, S., Wilkinson, D.J., 2010. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core (Release 1 Candidate). Nature Precedings.

Carlisle, D., Ion, P., Miner, R., Ausbrooks, R., Buswell, S., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Rowley, C., Sargent, M., Smith, B., Soiffer, N., Sutor, R., Watt, S., 2010. Mathematical Markup Language (MathML) Version 3.0, W3C Recommendation 21 October 2010.

CellML

Maintained by: Auckland Bioengineering Institute, University of Auckland

Latest version: v1.1

Availability: http://www.cellml.org/specifications/cellml_1.1

Description

Developed out of the cardiac modelling community, CellML (Lloyd et al., 2004) is a modelling markup language that aims to cover a range of biological phenomenon, primarily cell-function. CellML was developed to address the lack of standards for describing cellular function and to provide unambiguous representations of models. One of the key motivating factors is in publishing models to research communities. The authors identified that because of the lack of rigour and standards in the publishing process, models could not be easily validated. Errors are commonly introduced when publishing models in journal texts and computational implementations are commonly targeted at specific software frameworks and tools, making the models themselves less portable. This poses problems when sharing with researchers who are unfamiliar with the modelling methodologies, frameworks, and tools others may have used.

Like SBML, CellML utilises Content MathML to describe systems modelled using mathematical equations. While it does not explicitly restrict the allowed vocabulary in the same fashion as SBML, existing tools also only support ODEs and DAEs, and there is a recommended "CellML subset" of MathML. CellML is designed to be modular in that encapsulated models (possibly of different scales) can be linked together through public and private interfaces. This allows multiple models whose variables might refer to the same entity to be logically linked. This component-based approach allows reuse of whole models or parts of models described with CellML markup.

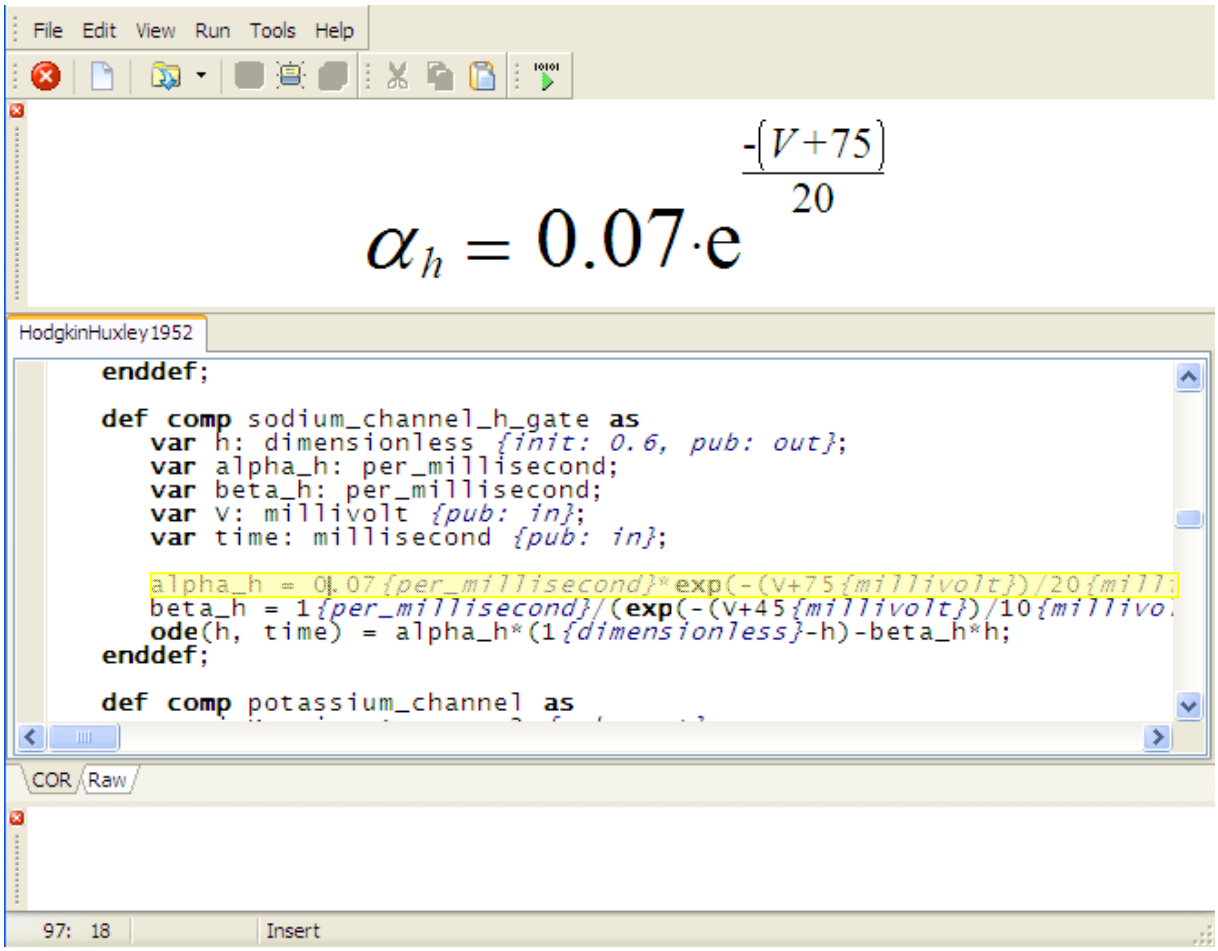

Figure 2: Cellular Open Resource (COR) (Garny et al., 2003), a software environment for modelling with CellML, in editorial mode. The mathematical equation shown in the top pane is rendered from the compact-form CellML markup highlighted in the middle pane.

There are a growing number of CellML tools (Garny et al., 2008), and a popular model repository (http://models.cellml.org). CellML has also been used to provide validated model descriptions alongside journal paper publications (Nickerson et al., 2008).

References

Lloyd, C.M., Halstead, M.D.B., Nielsen, P.F., 2004. CellML: its future, present and past. Progress in Biophysics and Molecular Biology 85, 433 – 450. Modelling Cellular and Tissue Function.

Garny, A., Kohl, P., Noble, D., 2003. Cellular open resource (COR): a public CellML based environment for modeling biological function. I. J. Bifurcation and Chaos 13, 3579–3590.

Garny, A., Nickerson, D.P., Cooper, J., Santos, R.W., Miller, A.K., McKeever, S., Nielsen, P.M.F., Hunter, P.J., 2008. CellML and associated tools and techniques. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366, 3017–3043.

Nickerson, D.P., Corrias, A., Buist, M.L., 2008. Reference descriptions of cellular electrophysiology models. Bioinformatics 24, 1112-1114.

FieldML

Maintained by: IUPS Physiome Project

Latest version: v0.3 (beta)

Availability: http://www.fieldml.org/

Description

FieldML is a markup language primarily for modelling physiological structures, and their physics, described as abstract fields representing the variation in some quantity over a particular domain (Christie et al. 2009). These quantities will typically represent some physical state, and may be simple scalars or more complex structures such as vectors or tensors. A basic field describes the domain itself, but the language also includes a rich set of operators, in part utilising MathML, for defining new fields as functions of other fields. This allows the representation of partial differential equations (PDEs) which may be solved by, for example, the finite element method (FEM) to simulate spatially varying behaviour, such as the diffusion of chemicals or electrical charge, or the mechanical motion of an organ.

FieldML is also being designed to complement CellML, allowing CellML models to be instantiated at all points within a domain, with field values at those points becoming inputs and outputs to the CellML models. This combination can represent models such as reaction-diffusion equations, with CellML defining the reaction source terms, and FieldML defining the geometry and diffusion properties.

FieldML is undergoing active development within other projects, notably the VPH project euHeart, but is still at early beta status, with frequent changes. It is thus too much in flux to depend upon within TUMOR.

References

Christie, G.R., Nielsen, P.M., Blackett, S.A., Bradley, C.P., Hunter, P.J., 2009. FieldML: concepts and implementation. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367, 1869–1884.

*insilico*ML

Maintained by: Physiome.jp / Worldwide Integrative Biomedical Research Cooperation

Latest version: v1.0

Availability: http://www.physiome.jp/insilicoml/index.html

Description

*insilico*ML (ISML) is a markup language for describing biological models developed out of the Japanese Physiome project (Asai et al., 2008; Suzuki et al., 2008, 2009). Like CellML, it was developed to be modular and it has many parallels with CellML. The authors designed ISML and its associated tools to facilitate conversion to multiple formats including CellML, SBML, C++ source code, and LaTeX. ISML supports a range of mathematical models such as those described with ODEs and PDEs, and agent-based models that utilise conditional constructs.

ISML represents a biological system in a similar fashion to CellML and SBML. It models a system as an aggregate of modules, where each module corresponds to an entity with state and a corresponding mathematical implementation. The implementation details how the states change in reaction to specific events and to the progression of time. Signalling and communication between modules is represented with graph-like edges that link input and output interfaces of modules termed ports. These edges enable the communication of

physical quantities representing state values. Ultimately by structuring biophysical models in such a way, models can be represented as graphs, hierarchies and independent modules.

ISML defines a concept called *capsulation*. This is where multiple modules can be packaged together, essentially encapsulating them in a larger capsule module. Capsule modules, like other modules, also possess input and output ports. To create logical connections between the capsules ports with the encapsulated modules, a specially defined edge called a forwarding edge links the capsule ports with internal module ports. Capsulation leads to hierarchical representations of componential models.

There are a number of parallels between ISML and CellML. For example, ISML modules and physical quantities correspond to CellML components and variables respectively. However the parallels only run so far. For example, although CellML connections between public/private interfaces are structurally similar to ISML edges and ports, in CellML the purpose of connections is to link entities of semantically equivalent variables. Because of this, the connections themselves do not model any directionality. In ISML, edges between ports have explicit direction inputs to outputs. Additionally, ISML edges have specific operational types attached to them, labelled with a verb or verb phrase describing the functional relationship. ISML modules also have a defined set of types such as functional units, containers, capsules (already discussed), and templates. The definitions of each of the ISML types are not discussed in this paper, but are extensively described in Asai et al. (2008) and Suzuki et al. (2008, 2009).

References

Asai, Y., Suzuki, Y., Kido, Y., Oka, H., Heien, E., Nakanishi, M., Urai, T., Hagihara, K., Kurachi, Y., Nomura, T., 2008. Specifications of insilicoml 1.0: a multilevel biophysical model description language. J Physiol Sci 58, 447–58.

Suzuki, Y., Asai, Y., Kawazu, T., Nakanishi, M., Taniguchi, Y., Heien, E., Hagihara, K., Kurachi, Y., Nomura, T., 2008. A platform for in silico mod- eling of physiological systems ii. cellml compatibility and other extended capabilities, in: Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, pp. 573 –576.

Suzuki, Y., Asai, Y., Oka, H., Heien, E., Urai, T., Okamoto, T., Yumikura, Y., Tominaga, K., Kido, Y., Nakanishi, M., Hagihara, K., Kurachi, Y., Nomura, T., 2009. A platform for in silico modeling of physiological sys- tems iii, in: Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pp. 2803 –2806.

Discussion

The reviewed markup languages serve as excellent examples of how to tackle biological modelling with markup. However we feel that, although appropriate for describing a wide range of models, they are limited in expressiveness when considering more complex phenomenon found in the cancer modelling literature. Each of the markup languages reviewed describes models based mainly on solving mathematical formulae, however other modelling techniques include more algorithmic approaches. Control flow constructs, such as loops and conditional behaviours, data type collections, such as arrays and matrices, and domain-specific data objects, would give a markup language more expressive power, especially where models are developed using an *in silico* methodology rather than a pure mathematical approach.

As a very generic example, consider a model that may be based on finite-state machines (FSMs) whereby each biological cells internal state is determined by events external to itself. States, transitions, conditions, and actions (all fundamental elements of FSMs) cannot be coded using any of SBML, CellML, FieldML or ISML. The definition of FSMs would require the use of formal logic that is not considered in any of these markup languages. We submit that the introduction of such constructs, which are commonly found in programming languages, would compliment pure mathematical modelling.

We do not propose that such an expansion of mathematical notation is undertaken to produce a general-purpose modelling markup. Using general-purpose languages for describing biological processes could introduce ambiguity in how models are developed and described. We also submit that the mathematics and computational descriptions should be encapsulated in domain-specific components, and consequently models described using domain-specific markup. Creating a language that is specific to computational cancer modelling, and using terminology that will be familiar to the field, will allow modellers to more easily on the scientific questions at hand rather than spending significant effort on learning how to use yet another general purpose programming language. Models will also become more modular allowing component reuse by restricting the concepts and cancer terminology used in the markup descriptions. General-purpose languages are not suited to domain-specific modularisation as the specification of interfaces between model components is left entirely open to ambiguous definition.

Each of the currently available markup languages addresses biological modelling by encapsulating the mathematics that underpins each component part of a model. However it should be noted that the expressiveness in each respective modelling language lacks somewhat when applied to more complex scenarios. For example, each modelling language bases its mathematical descriptions on MathML. While MathML consists of a mature vocabulary, it does not provide any way of expressing logic and control flow or complex data constructs. The models based on markup using MathML are typically simulated through solving ODEs and DAEs. SBML is a very specialised language and represents models through describing low-level molecular components and their relationships with each other. CellML relies on declarative mathematics that is interpreted and processed by numerical solvers, mainly to model biological cell function. The domain concepts in CellML are decoupled from the language and included as metadata. FieldML is being developed as a language to compliment CellML in modelling physiological structures based on geometric meshes and fields. ISML is similar to CellML is its application to a wide range of biology, and also demonstrates multi-scale application. Algorithmic and cellular automata-based cancer models, such as in the top-down approach of the Oncosimulator (Stamatakos et al, 2007), and agent-based molecular modelling from the bottom-up, cannot be expressed in any of the currently available markup languages, let alone before considering any hybrid top-down-bottom-up composite models.

The generic application target of these markup languages also hinders their adoption and usage in the cancer modelling community. SBML is a specialised language that describes molecular components and their relationships with each other. CellML depends on declarative mathematics that is processed by numerical solvers, mainly to model cell function, and the domain concepts in CellML are decoupled from the language as metadata annotations. FieldML is still in its infancy, and while it will add a spatial element to CellML descriptions, it is however limited to continuous models of behaviour, being unable to represent the agent-based approaches described above. ISML is similar to CellML in its application to a wide range of biology, and also demonstrates multi-scale application, but again in a very generic fashion. Although SBML, CellML and FieldML each addresses specific aspects of modelling biology, none of these can capture the multi-scale aspect of cancer modelling, and ISML is not domain-specific enough to capture cancer modelling terminology. CellML and ISML both address the modularisation of models, but neither facilitates capturing domain-specific cancer model definitions. As we believe the existing state-of-the-art modelling languages do not fulfil the remit of model interoperability for the cancer-modelling domain, we propose the development of a new markup language as part of the work falling under WP4, and the design of which described in the following sections.

References

Stamatakos, G., Dionysiou, D., Graf, N., Sofra, N., Desmedt, C., Hoppe, A., Uzunoglu, N., Tsiknakis, M., 2007. The “oncosimulator”: a multilevel, clinically oriented simulation system of tumor growth and organism re- sponse to therapeutic schemes. towards the clinical

evaluation of in silico oncology, in: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pp. 6628 –6631.

4 Existing Reusable Markup Vocabularies

Although there are a number of existing modelling markup languages, as described in the previous section, their application to the cancer-modelling domain is limited, as already discussed. In developing a new markup language, we will firstly review the currently available markup vocabularies that could be integrated into a new markup specification. The set of markup languages described in this section is not exhaustive, but aims to provide a review of the main kinds of markup that could be integrated to allow various functionality without having to 'redefine the wheel'. We review markup languages relating to mathematical and computational descriptions, resource description/curation, medical and clinical data, and workflow composition, each of which has the potential to be integrated for use within the TUMOR project's own markup.

MathML

Maintained by: World Wide Web Consortium

Latest version: v3.0

Availability: http://www.w3.org/TR/2010/REC-MathML3-20101021/

Description

MathML is a markup specification to describe mathematics for machine-to-machine interpretation and for display on Web pages (Carlisle et al, 2010). Developed and maintained by the W3C, it provides an unambiguous language in which to describe mathematical formulae that can be interpreted for numerical processing or to render as human-consumable content in alternative ways to visual display. Tim Berners-Lee described the potential for MathML as follows:

> *"MathML will make the Web even better for educational, scientific and technical materials. It also has the potential to make mathematics accessible to those with visual disabilities. It will allow mathematical content to be reused and exchanged with technical computing systems for further manipulation."* (Ion et al, 2002)

MathML defines two subsets of markup – (1) Presentation MathML, and (2) Content MathML.

Presentation MathML is designed solely for rendering mathematical expressions in Web pages, embedding the markup code into HTML pages. What distinguishes MathML from other mathematical markup languages, such as TeX/LaTeX and SGML mathematical markup defined in ISO 12083 Mathematics DTD, is that it additionally provides markup for describing the semantics of the mathematics, where the aforementioned alternatives only concentrate on document display rendering. Content MathML provides an alternative form of markup to Presentation MathML in order to capture the semantics of the mathematics being described. Table 1 illustrates an example of Presentation MathML versus Content MathML.

As discussed in the previous section on Markup Languages for Biological Modelling, MathML has been used as the basis for the current state-of-the-art biomodelling languages including each of SBML, CellML, and ISML.

Within the TUMOR project, models will be published as executable files and source code, however there is potential to augment these with MathML descriptions of the mathematical components of published models where applicable.

<mrow> <mrow> <msup> <mi>x</mi> <mn>2</mn> </msup> <mo>+</mo> <mrow> <mn>4</mn> <mo>⁢</mo> <mi>x</mi> </mrow> <mo>+</mo> <mn>4</mn> </mrow> <mo>=</mo> <mn>0</mn> </mrow>	<apply> <plus/> <apply> <power/> <ci>x</ci> <cn>2</cn> </apply> <apply> <times/> <cn>4</cn> <ci>x</ci> </apply> <cn>4</cn> </apply>

Table 1 Example contrasting Presentation MathML (left) against Content MathML (right). Both examples represent the equation $x^2 + 4x + 4 = 0$ (MathML, 2011).

Strengths

- Many biological models are based on solving numerical equations which can be expressed in MathML
- Demonstrated usage in state-of-the-art modelling markup languages

Weaknesses

- Is not expressive enough to describe all mathematical models
- Not applicable to all kinds of modelling, where some models use computational logic.

References

David Carlisle, Patrick Ion and Robert Miner (eds.). 2010. Mathematical Markup Language (MathML) Version 3.0 W3C Recommendation 21 October 2010. [Online] http://www.w3.org/TR/2010/REC-MathML3-20101021/

Patrick Ion (ed.), Stephen Buswell and Robert S. Sutor. 2002. Mathematical Markup Language (MathML) Frequently Asked Questions (FAQ). [Oniline] http://www.w3.org/Math/mathml-faq.html

Mathematical Markup Language (MathML) What is MathML? [Online] http://www.w3.org/Math/whatIsMathML.html

Stephen M. Watt and Xuehong Li. 1999. Examples of MathML. SIGSAM Bull. 33, 1 (March 1999), 1-4. DOI:10.1145/329984.329985

Job Submission Description Language

Maintained by: Open Grid Forum

Latest version: v1.0

Availability: http://forge.gridforum.org/sf/projects/jsdl-wg

Description

The Job Submission Description Language (JSDL) is an open standard for describing computational job executions (Anjomshoaa et al, 2005). Originally developed by the Global

Grid Forum (GGF), it is now maintained by the Open Grid Forum (OGF), due to a merger between the GGF and the Enterprise Grid Alliance. JSDL job descriptions are not intended to describe running computational jobs, but rather are used statically to declare the requirements, configuration, and interfaces that are needed to describe a computational application. Typically, applications described in JSDL are run in Grid Computing (Foster et al, 2001) environments, however JSDL can be used to describe any computational application regardless of underlying infrastructure. Grid computing systems are typically middleware-oriented, and although JSDL is developed as a standard, many Grid middleware do not directly support JSDL. However, as an open standard this means that translation between JSDL descriptions and specific implementations is straightforward. For example, the g-Eclipse workbench (Gjermundrod et al, 2008) provides facility to translate JSDL, including workflows composed of multiple JSDL descriptions (Johnson et al, 2009), on-the-fly to other formats for whatever available Grid middleware the workbench is configured to connect to.

JSDL allows computational applications to be described with a set of properties including (but not limited to) the following:

- General properties such as an application name and path to an executable binary application
- Application-specific settings such as standard input/output/error files and environment variables
- Data-staging paths, such as where files that the application needs as input are located and where output files are written to
- Resources, such as the specific hardware and software requirements that the application needs to be able to run.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl">
   <JobDescription>
      <JobIdentification>
         <JobName>runModel</JobName>
      </JobIdentification>
      <Application>
         <ApplicationName>runModel</ApplicationName>
      </Application>
      <Resources>
         <IndividualDiskSpace>
            <LowerBoundedRange>20000.0000000000</LowerBoundedRange>
         </IndividualDiskSpace>
      </Resources>
      <DataStaging name="inputdata-001">
         <FileName>inputdata.txt</FileName>
         <CreationFlag>overwrite</CreationFlag>
         <DeleteOnTermination>true</DeleteOnTermination>
      </DataStaging>
   </JobDescription>
</JobDefinition>
```

Table 2 Example JSDL file describing an application, 'runModel' with a default input file 'inputdata.txt' and the resource requirement of a minimum of 20,000kb of disk space.

Table 2 shows an example JSDL file describing a simple computational job. The resource requirements can be a lot more specific to include details such as the required operating system, processor architecture, processor speeds and memory requirements. The details of the JSDL can be found fully described in the OGF specification by Anjomshoaa et al (2005).

To our knowledge, JSDL has never been incorporated into any biological modelling markup as the current state-of-the-art languages explicitly describe the internal model structure and functionality through other means, typically using MathML. However, JSDL has been used as a standard for publishing applications to an online repository for the UK National Grid Service (NGS), as described in Meredith et al (2007), which demonstrates its potential use in to other domains requiring digital curation.

Using JSDL within the TUMOR project would bring several benefits. Firstly, it potentially provides a mechanism for interfacing published models with whatever execution environment is used to run models, both on the US side (within CViT via the Computational Modeling Execution Framework, CMEF) and the EU side (within the TUMOR workflow and execution environment). JSDL being XML based makes it more easily translatable to formats required by whatever execution environment is used. Secondly, and most importantly within the scope of the TUMOR project, JSDL allows the specification of hardware and software requirements. Where we envisage implementations of models stored in the TUMOR repository to have specific platform needs, metadata describing the runtime environment is essential. Finally, JSDL allows the specification of input parameters (via the command line) and input and output files. This information is again implementation specific, and therefore is required in order to be able to run simulations using published models.

Strengths

- Assists in specification of computational hardware/software requirements, which is essential for WP5 Integrated Workflow Execution Environment
- Industry standard used in various execution environments and with good tools support

Weaknesses

- Does not provide internal descriptions of the executable file(s) in question
- Does not link to biological domain concepts
- No demonstrated use in currently available biological markup languages.

References

A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, "Job Submission Description Language (JSDL) Specification V1.0," Open Grid Forum, GFD 56 2005

Ian T. Foster. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing (Euro-Par '01), Rizos Sakellariou, John Keane, John R. Gurd, and Len Freeman (Eds.). Springer-Verlag, London, UK, 1-4.

H. Gjermundrod, M. D. Dikaiakos, M. Stumpert, P. Wolniewicz, and H. Kornmayer. 2008. g-Eclipse - an integrated framework to access and maintain Grid resources. In Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing (GRID '08). IEEE Computer Society, Washington, DC, USA, 57-64. DOI:10.1109/GRID.2008.4662783

Johnson, D., Meacham, K. E. and Kornmayer, H. (2009) A Middleware Independent Grid Workflow Builder for Scientific Applications. In: 2009 5th IEEE International Conference on e-Science Workshops, 2009, Oxford.

D. Meredith, M. Maniopoulou, A. Richards, M. Mineter: A JSDL Application Repository and Artefact Sharing Portal for Heterogeneous Grids and the NGS, Proceedings of the UK e-Science All Hands Meeting 2007, Nottingham, UK, 10th-13th September 2007, pp 110-118, ISBN 978-0-9553988-3-4

Dublin Core

Maintained by: Dublin Core Metadata Initiative

Latest version: ISO 15836:2009

Availability: http://dublincore.org/documents/2010/10/11/dcmi-terms/

Description

A long established metadata vocabulary for annotating generic electronic resources is a set of elements called Dublin Core. Today maintained and promoted as a standard for metadata by the Dublin Core Metadata Initiative (DCMI), the element set is widely used in various topic areas that utilise digital repositories including in biological modelling. The original motivation for Dublin Core was the need to annotate web resources with metadata to more easily catalogue and make available online collections of documents. Originally designed to be author-populated fields, the implications for resource collection providers (i.e. museums and libraries, online or otherwise) were obvious – digital curation of resources by adding metadata to them would make searching and retrieving resources easier via semantic search and matching. The original authors identify that there are domain-specific elements that could be used to annotate resources to a finer level of detail, but also acknowledge it is a generic approach that is required to enable widespread adoption of the element set. The approach to identifying Dublin Core's element set was simple: all resources share common metadata.

The Dublin Core Metadata Element Set (v1.1) is comprised of 15 metadata elements [cite]:

1. Title – the name of the resource given by the creator or publisher
2. Creator – the person or organisation responsible for creating the intellectual content of the resource at hand
3. Subject – the topic of the resource. Perhaps a list of keywords, or phrases describing the resource. This may be a controlled vocabulary and the definition of which is left to the implementation
4. Description – a textual description of the resource. This may be an abstract or a textual description for non-textual content
5. Publisher – the person or organisation responsible for making the intellectual content of the resource at hand available. Note that this may be a different entity to the creator
6. Contributor – persons or organisations that, in addition to the creator, may have contributed to creating the intellectual content of the resource at hand
7. Date – the date the resource was published (specific to this particular version of the resource)
8. Type – a categorisation of the resource chosen from a finite enumeration defined elsewhere by DCMI
9. Format – a specification of the data format of the resource chosen from a finite enumeration defined elsewhere by DCMI
10. Identifier – a unique string or number to identify the resource with. The uniqueness and identification scheme is determined by the implementation. Typical schemes might be URLs or ISBNs
11. Source – where applicable, a description of where the resource was originally derived from
12. Language – the language in which the resource is expressed in

13. Relation – abstract relationship descriptions to other resources
14. Coverage – spatio-temporal characteristic of the resource
15. Rights – licensing or rights management via linking to an appropriate descriptor of the legal requirements associated with the resource.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description
rdf:about="http://media.example.com/audio/guide.ra">
      <dc:creator>Rose Bush</dc:creator>
      <dc:title>A Guide to Growing Roses</dc:title>
      <dc:description>Describes process for planting and nurturing
different kinds of rose bushes.</dc:description>
      <dc:date>2001-01-20</dc:date>
   </rdf:Description>
</rdf:RDF>
```

Table 3 An example of Dublin Core metadata markup (Hillmann, 2005).

Table 3 shows an example of Dublin Core markup taken from Hillmann (2005). It describes an audio file, annotating metadata describing the creator, a title of the resource, a brief description, and a publication date. Dublin Core has been used to provide the basic descriptive parts of biological model metadata as it allows more meaningful search functionality in online digital model repositories.

The TUMOR model repository will provide facilities for publication, search, and retrieval of models, and Dublin Core would provide more meaningful standardised semantic structure to the metadata attached to published models.

Strengths

- Standardised metadata for generic resources
- Widely used in all domains, including existing biological modelling markup standards

Weaknesses

- Typically each element is expressed as free text fields. This means that there are no standard vocabularies for the element contents
- Does not incorporate domain-specific elements in the markup standard.

References

S. Weibel , J. Kunze , C. Lagoze , M. Wolf, Dublin Core Metadata for Resource Discovery, RFC Editor, 1998

Weibel, S., "The Dublin Core: A simple content description format for electronic resources," NFAIS Newsletter, 40 (7), pp. 117--119, 1998

Diane Hillmann, “Using Dublin Core.” 2005. [Online] http://dublincore.org/documents/usageguide/

RDF vCard

Maintained by: World Wide Web Consortium; Internet Engineering Task Force

Latest version: W3C Member Submission 20 January 2010; RFC 2425, 2426

Availability: http://www.w3.org/Submission/2010/SUBM-vcard-rdf-20100120/

Description

Developed and maintained by the Internet Engineering Task Force (IETF), the vCard standard defines metadata for describing people and organisations (Dawson and Howes, 1998). The main idea is that vCards are synonymous with physical business cards to provide specific information related to a particular person or organisation that can be used as a unique identifier in a meaningful way (without the use of a universally unique identifier). vCards typically describe an entity using optional elements including: a name, organisation, roles within the organisation, telephone numbers, postal addresses, email addresses etc. They can also refer to multimedia and web resources where appropriate. For example, a vCard might describe a fictional person as follows:

```
BEGIN:VCARD
  VERSION:2.1
  N:Gump;Forrest
  FN:Forrest Gump
  ORG:Bubba Gump Shrimp Co.
  TITLE:Shrimp Man
  TEL;WORK;VOICE:(111) 555-1212
  TEL;HOME;VOICE:(404) 555-1212
  ADR;WORK:;;100 Waters Edge;Baytown;LA;30314;United States of America
  LABEL;WORK;ENCODING=QUOTED-PRINTABLE:100 Waters Edge=0D=0ABaytown, LA
30314=0D=0AUnited States of America
  ADR;HOME:;;42 Plantation St.;Baytown;LA;30314;United States of America
  LABEL;HOME;ENCODING=QUOTED-PRINTABLE:42 Plantation St.=0D=0ABaytown,
  LA 30314=0D=0AUnited States of America
  EMAIL;PREF;INTERNET:forrestgump@example.com
  REV:20080424T195243Z
END:VCARD
```

Table 4 An example vCard encoded in RFC 2426.

As illustrated by Table 4, the raw vCard format as defined and maintained by the IETF is not an XML format. However there is an RDF encoding for RFC 2426 vCards that provides equivalent functionality (Iannella, 2001). For example, the previous example encoded in its RDF equivalent would be as illustrated in Table 5.

The RDF vCard encoding vocabulary has been utilised in some of the existing biological modelling languages to describe authors and organisational relationships to model creators and publishers, typically in combination with the appropriate Dublin Core elements.

Actors within the TUMOR repository (i.e. Clinicians, researchers, scientists, modellers, managers etc.) need to be represented by more than just a username. vCard would provide a metadata set to describe people more fully and a means to efficient provenance management of data uploaded to the repository.

Strengths

- Maintained by universally accepted Internet standards organisations (IETF and W3C for RDF encoding)
- Widely used in all domains, including industrial/commercial applications

Weaknesses

- Very verbose
- More business oriented, so unclear as to whether additional metadata is needed when representing clinicians and researchers.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:v="http://www.w3.org/2006/vcard/ns#">
<v:VCard rdf:about = "http://example.com/me/forestgump">
   <v:fn>Forest Gump</v:fn>
   <v:tel>
      <rdf:Description>
         <rdf:value>(404) 555-1212</rdf:value>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Home"/>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Voice"/>
      </rdf:Description>
   </v:tel>
   <v:tel>
      <rdf:Description>
         <rdf:value>(111) 555-1212</rdf:value>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Work"/>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Voice"/>
      </rdf:Description>
   </v:tel>
   <v:email rdf:resource="mailto:forrestgump@example.com"/>
   <v:adr>
      <rdf:Description>
         <v:street-address>42 Plantation St.</v:street-address>
         <v:locality>Baytown, LA</v:locality>
         <v:postal-code>30314</v:postal-code>
         <v:country-name>United States of America</v:country-name>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Home"/>
      </rdf:Description>
   </v:adr>
   <v:adr>
      <rdf:Description>
         <v:street-address>100 Waters Edge </v:street-address>
         <v:locality>Baytown, LA</v:locality>
         <v:postal-code>30314</v:postal-code>
         <v:country-name>United States of America </v:country-name>
         <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Work"/>
      </rdf:Description>
   </v:adr>
</v:VCard>
</rdf:RDF>
```

Table 5 The RDF encoding of the vCard RFC 2426 example.

References

F. Dawson and T. Howes. 1998. RFC 2426: vCard MIME directory profile, IETF.

R. Iannella. Representing vCard Objects in RDF/XML, 2001.

FOAF

Maintained by: FOAF Project

Latest version: v0.98

Availability: http://xmlns.com/foaf/spec/20100809.html

Description

The FOAF (Friend of a Friend) project defines an RDF-based vocabulary to describe people and their relationship with artefacts and other people on the Web (Brickley and Miller, 2005). Initially developed in 2000 by Dan Brickley and Libby Miller, FOAF is maintained as an open

source project at http://www.foaf-project.org. The main idea behind FOAF is being able to publish representations of how one particular person is connected to a variety of resources. Conceptually, this provides a way of building a network of information of not only how people are connected, but also how Web resources are connected. The authors explain FOAF with a fictional example:

> *"**Dan** lives in **Zetland road, Bristol, UK** with **Libby** and **Craig**. **Dan**'s email address is **danbri@w3.org**. Libby's email address is **libby.miller@bris.ac.uk**. **Craig**'s is **craig@netgates.co.uk**. **Dan** and **Libby** work for an organisation called **"ILRT"** whose website is at **http://ilrt.org/**. **Craig** works for **"Netgates"**, an organisation whose website is at **http://www.netgates.co.uk/**. **Craig**'s wife **Liz** lives in **Bristol** with **Kathleen**. **Kathleen** and **Liz** also work at **"Netgates"**. **Damian** lives in **London**. **Martin** knows **Craig**, **Damian**, **Dan** and **Libby** quite well. **Martin** lives in **Bristol** and has an email address of **m.l.poulter@bristol.ac.uk**. (etc...)"* (Brickley and Miller, 2008)

In this example, the authors try to illustrate how by using short factual sentences information can be inferred about people and organisations. By analysing each of these short factoids, we can make connections between people, places and organisation – e.g. when we consider connections between the named people in the example, we might infer that Dan works with Libby, Libby knows Craig, Craig is married to Liz, and so on. This builds a web of information where Dan is connected to Liz, be it indirectly.

```
<foaf:Person rdf:about="#davidjohnson"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
   <foaf:name>David Johnson</foaf:name>
   <foaf:homepage rdf:resource="
http://www.comlab.ox.ac.uk/people/david.johnson/" />
   <foaf:img rdf:resource="/images/me.jpg" />
</foaf:Person>
```

Table 6 A simple example of a FOAF document describing a person, linking them to a Web page and an image file.

A simple example of a FOAF document is shown in Table 6, however it should be noted that the detail in which FOAF can describe people and resources could go into much greater depth. The full details of the current FOAF vocabulary are published by Brickley and Miller (2010), and will not be detailed in this deliverable.

FOAF links people and resources with other people and online resources. By using FOAF within TUMOR, we may be able to infer relationships between the different actors using the repository. Not only this, we would be able to use FOAF to express specific relationships between published models and external resources such as references to published journal papers or websites.

Strengths

- Can be used to map networks of relationships between actors and resources

Weaknesses

- Does not provide (on its own) detailed metadata for the elements within the networks described.

References

D. Brickley and L. Miller, " FOAF Vocabulary Specification 0.98 Namespace Document 9 August 2010 - Marco Polo Edition," http://xmlns.com/foaf/spec/20100809.html, 2010.

D. Brickley and L. Miller, "Introducing FOAF", http://www.foaf-project.org/original-intro, 2008.

Simulation Experiment Description Markup Language

Maintained by: SED-ML Community

Latest version: Level 1 RC1

Availability: http://www.biomodels.net/sed-ml/

Description

The modelling markup languages described above form part of a solution to the problem of sharing and reuse of computational models of biology, allowing the encoding of model structure and mathematics in a widely supported standard format. However, simply having a verified representation of the equations in a computer-readable format is not sufficient for supporting reliable and efficient reuse. The functional characteristics of the model also have to be known, which requires a description of simulations run using the model(s), and expected outputs.

To address this, firstly a set of guidelines called the Minimum Information About a Simulation Experiment (MIASE) has been developed, setting out in general terms what details are required in order to be able to reproduce a simulation. It covers information about the simulation settings, including information about the models, changes on them, simulation settings applied to the models and output definitions.

SED-ML is an XML format that enables the storage and exchange of part of the information required to implement the MIASE guidelines. SED-ML is independent of the formats used to encode the models – as long as they are expressed in XML – and it is independent of the software tools used to run the simulations. Several test implementations are being developed to benchmark SED-ML on simple cases, and pave the way to a more complete support of MIASE.

It is thus still in the early stages of development, and not yet suitable for direct use in TUMOR. However, there is strong backing from the SBML community, and so future projects should consider it.

Strengths

- Independent of modelling language for describing how to reproduce simulations

Weaknesses

- Not a mature standard and has yet to be demonstrated with widespread adoption and use.

References

Dagmar Köhn and Nicolas Le Novère, “SED-ML – An XML Format for the Implementation of the MIASE Guidelines,” Computational Methods in Systems Biology, Lecture Notes in Computer Science, 2008, Volume 5307/2008, 176-190, DOI: 10.1007/978-3-540-88562-7_15

HL7 Clinical Document Architecture

Maintained by: Health Level 7 International

Latest version: v3

Availability: http://www.hl7.org/implement/standards/cda.cfm

Description

The Clinical Document Architecture (CDA) is a markup standard for describing the structure and semantics of clinical documents. Developed by Health Level Seven (HL7), an organisation that develops health informatics interoperability standards, CDA forms part of

the wider HL7 version 3 standard that additionally includes standards for the general implementation model and messaging framework that HP7-compliant systems adhere to. CDA specifies a number of required characteristics:

- Persistence – A clinical document continues to exist in an unaltered state, for a time period defined by local and regulatory requirements
- Stewardship – A clinical document is maintained by an organization entrusted with its care
- Potential for authentication - A clinical document is an assemblage of information that is intended to be legally authenticated
- Context - A clinical document establishes the default context for its contents
- Wholeness - Authentication of a clinical document applies to the whole and does not apply to portions of the document without the full context of the document
- Human readability – A clinical document is human readable.

'Clinical documents,' as defined by CDA, are not intended to be whole medical records of a single patient. They serve as a way of transmitting clinical data, and therefore are only expected to hold specific reports relating to individual instances relating to a patient's record. An examples of the structure of a CDA document is shown in Table 7.

```
<ClinicalDocument>
   ... CDA Header ...
   <StructuredBody>
      <section>
         <text>...</text>
         <Observation>
            ...
         </Observation>
         <Observation>
            <reference>
               <ExternalObservation>
                  ...
               </ExternalObservation>
            </reference>
         </Observation>
      </section>
      <section>
         <section>
            ...
         </section>
      </section>
   </StructuredBody>
</ClinicalDocument>
```

Table 7 The main elements of a CDA document

CDA specifies that its primary use is for data exchange – not for creation and management of clinical documents. This means that in typical usage, clinical documents expressed in CDA markup are not stored in digital repositories.

Within the TUMOR project's infrastructure, it may be required to transmit clinical data between clinical databases and the execution environment to run and validate computational cancer models. CDA would provide a standard format for transmission of clinical data within TUMOR.

Strengths

- Industry accepted standard for clinical data exchange

- Generic enough to be applied to a wide range of types of clinical data

Weaknesses

- Does not provide markup for standalone data storage and management
- CDA is not intended to contain whole patient's medical records.

References

Robert H Dolin, Liora Alschuler, Calvin Beebe, Paul V Biron, Sandra Lee Boyer, Daniel Essin, Elliot Kimber, Tom Lincoln, and John E Mattison, "The HL7 Clinical Document Architecture," The Practice of Informatics, JAMIA 2001;8:552-569 doi:10.1136/jamia.2001.0080552

Medical Markup Language

Maintained by: MedXML Consortium

Latest version: v3.0

Availability: http://www.medxml.net/E_mml30/

Description

The Medical Markup Language (MML) is a language for describing different kinds of medical data, including clinical patient data, for a range of different medical fields (Kenji et al, 2000). Developed by the Japan Association for Medical Informatics "Electronic Health Record Research Group" to address interoperability when exchanging medical data between different hospitals and medical information providers. MML is split into nine modules that each addresses a specific type of medical data. These are:

1. Patient information
2. Health insurance information
3. Diagnosis information
4. Life style information
5. Basic medical information
6. Particular information at the time of first visit
7. Progress course information
8. Surgery information
9. Clinical summary information.

Each module is defined with its own XML namespace, and the authors designed MML to allow the linking of different kinds of documents through unique document identifiers. This would allow, for example, one particular patient information document to be linked to a diagnosis information document.

In recognition of the HL7 CDA becoming an industry accepted standard for exchange of clinical data, the authors of MML developed their latest version to be compatible with transmission as HL7 CDA payloads. By basing MML on HL7 CDA, this allows MML to be transported in accordance with the HL7 specification, but also allows MML payloads to also be represented as stand-alone documents.

Apart from transmission of clinical data, a fully integrated environment that incorporates TUMOR's model repository and model execution environment may include infrastructure components for the actual management of clinical data. MML could then be used as a standard format for clinical data storage within TUMOR making it more easily accessible to run simulations.

Strengths

- Is a useful specification for storage of patient records and clinical data in an XML format
- Compatible with HL7 CDA messages making MML payloads transportable between HL7 compliant systems.

Weaknesses

- Unclear as to whether it is a widely accepted markup standard for medical data as it is developed and maintained by the Japan Association for Medical Informatics, which seems to target systems using the Japanese language.

References

Kenji Araki, Katsuhiro Ohashi, Shunji Yamazaki, Yasuyuki Hirose, Yoshinori Yamashita, Ryuichi Yamamoto, Kazushi Minagawa, Norihiro Sakamoto and Hiroyuki Yoshihara, "Medical Markup Language (MML) for XML-based Hospital Information Interchange," Journal of Medical Systems, Volume 24, Number 3, 195-211, DOI: 10.1023/A:1005595727426, 2000

MedXML Consortium, "MML Version 3.0 Specification." 2003. [Online] http://www.medxml.net/E_mml30/

XML Simple Conceptual Unified Flow Language

Maintained by: The Taverna Project

Latest version: v0.1 (alpha)

Availability: http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html

Description

Developed out of the UK myGrid project, the Simple Conceptual Unified Flow Language (Scufl) is a markup language for describing workflows. Initially conceived for creating bioinformatics workflows using the Taverna software package, Scufl describes executable applications (termed procesors), data sources, and their dependencies on each other and myGrid enacts workflows using the IT Innovation Enactment Engine. An XML version of the Scufl specification (XScufl) allows the definition of workflows using an XML-based vocabulary and a different enactment engine called Freefluo. Table 8 illustrates the markup for a trivial workflow that concatenates two strings.

To enable model composition, markup is needed to represent connections between component models as graphs, in a similar fashion to workflows. As one of the aims of TUMOR is to demonstrate transatlantic scenarios, the facilities to enable either transatlantic workflow composition or model coupling needs to be demonstrated. XScufl could potentially be used for either purpose.

Strengths

- Describes workflows in an abstract manner with an extensible language that can be adapted for other purposes than myGrid systems
- Scufl is extensively used by the UK eScience community

Weaknesses

- XScufl is only provided as an alpha status prototype and the specification is not currently actively maintained or being developed further.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:scufl xmlns:s="http://org.embl.ebi.escience/xscufl/0.1alpha"
version="0.2" log="0">
   <s:workflowdescription
lsid="urn:lsid:www.mygrid.org.uk:operation:BD8CRS09KB0" author="Tom Oinn"
title="Example of an alternate processor">
      Trivial workflow which will initially fail, retry twice then fall
over to the alternative specified for the FailingThing process.
   </s:workflowdescription>
   <s:processor name="FooString">
      <s:stringconstant>foo</s:stringconstant>
   </s:processor>
   <s:processor name="BarString">
      <s:stringconstant>bar</s:stringconstant>
   </s:processor>
   <s:processor name="FailingProcessor">
      <s:local maxretries="2" retrydelay="1000" retrybackoff="2.0">
         org.embl.ebi.escience.scuflworkers.java.TestAlwaysFailingProcessor
      </s:local>
      <s:alternate>
         <s:local>
            org.embl.ebi.escience.scuflworkers.java.StringConcat
         </s:local>
         <s:outputmap key="urgle" value="output" />
         <s:inputmap key="foo" value="string1" />
         <s:inputmap key="bar" value="string2" />
      </s:alternate>
   </s:processor>
   <s:link source="FooString:value" sink="FailingProcessor:foo" />
   <s:link source="BarString:value" sink="FailingProcessor:bar" />
   <s:link source="FailingProcessor:urgle" sink="out" />
   <s:sink name="out" />
</s:scufl>
```

Table 8 Exampe trivial workflow expressed in XScufl (Hoheisel, 2005)

References

Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Mark Greenwood, Carole Goble, Anil Wipat, Peter Li, and Tim Carver. 2004. Delivering web service coordination capability to users. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters (WWW Alt. '04). ACM, New York, NY, USA, 438-439. DOI:10.1145/1013367.1013514

T. Oinn (2004,Apr. 7), XScufl Language Reference [Online]. Available: http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html

Andreas Hoheisel, “XScufl.” 2005. [Online] http://www.gridworkflow.org/snips/gridworkflow/space/XScufl

Business Process Execution Language

Maintained by: OASIS

Latest version: v2.0

Availability: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Description

The Business Process Execution Language (BPEL) is an XML language for describing business process behaviour based on Web services. The BPEL workflow description language supports many of the features found in modern programming languages like flow

control, variables, concurrent execution, input and output, transaction scoping, compensation, and error handling. BPEL is layered on top of other Web technologies such as WSDL, XML Schema, XPath, XSLT, and WS Addressing.

Each BPEL document specifies the behaviour of a business process. BPEL processes often invoke Web services to perform functional tasks and can be either abstract or executable.

- Abstract processes are similar to library APIs: they describe what the process can do and its inputs and outputs but do not describe how anything gets done. Abstract processes are useful for describing a business process to another party that wants to use the process.
- Executable processes do the “heavy lifting” - they contain all of the execution steps that represent a cohesive unit of work.

Data in the BPEL is represented through XML types either the built-in ones or the types that can found in the XML Schemas imported in the BPEL document. Variables are then used to hold the data as is common in popular programming languages like Java. By assigning types to these variables it then becomes possible to validate both statically and dynamically that the data passed to and from a Web Service using SOAP messages are compatible with the parameters declared in the WSDL description of the service. Variables can be defined inside a scope and these scopes can be nested. Scopes not only prevent variable name clashes, but they also allow a BPEL execution engine to decide when the variables will no longer be needed and the, potentially large, data structures stored in a variable can be released.

A BPEL process typically consists of activities connected by links that perform the actual work in order to deliver the final outcome of the process. The path taken through the activities and their links is determined by many things, including the values of variables and the evaluation of expressions. The most important basic activity in BPEL is used to invoke web services. Each Web Service is represented as a partner in the BPEL parlance and each partner must define WSDL port types for each interface that is used in the BPEL process.

Invocation of a service can be either synchronous or asynchronous. Both are defined by the BPEL invoke construct. Invocations that give both input and output variables are executed synchronously and the BPEL workflow is blocked while the service executes. An example of such invocation is shown in Table 9.

```
<invoke name="executeQuery"
        partnerLink="SRV1PL"
        operation="executeQuery"
        inputVariable="query-in" outputVariable="result"/>
```

Table 9 Example of invoking a service.

For asynchronous execution, BPEL supports the notion of correlation sets that are used to associate replies to invocations with business process instances.

Every BPEL process is, in fact, a web service in its own right. The service can be invoked by sending a SOAP message to a BPEL engine. The BPEL primitive used to achieve this is a receive statement as shown in Table 10.

```
<receive name="Start"
         partnerLink="enactment" operation="doIt"
         variable="input" createInstance="yes"/>
```

Table 10 Sending a SOAP message to a BPEL engine.

Another important basic activity allows us to manipulate data stored in variables. This is done using assignments. An assignment consists of any number of copy statements that copy

data from a source to a target destination. Source destinations can be XML data given as a literal, the result of evaluating an expression, an XPath query that extracts data from some other variable, or the result of calling a procedure in a programming language, such as Java or JavaScript that can be incorporated using the BPEL extension mechanism.

```
<assign>
   <copy>
      <from>
         <literal>
            <message>Hello World</message>
         </literal>
      </from>
      <to variable="sayHello" part="parameters"/>
   </copy>
</assign>
```

Table 11 Assignments in BPEL.

Finally BPEL provides further control flow facilities in a straightforward manner. These include sequence, while, switch and pick structured activities. Sequence, while and switch have the conventional semantics, while Pick provides for non-deterministic choice.

Strengths

- Ability for reuse within TUMOR as workflow editor and execution using BPEL was developed under ACGT.

Weaknesses

- As BPEL is designed to orchestrate Web services, it might not be useful for execution infrastructures that do not expose such services.

References

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic I., & Weerawarana, S. (May 2003). Business process execution language for web services version 1.1 (Technical report)

5 Ontologies for the Clinical Medicine and Cancer Domain

In computer science, ontologies are a technique or technology used to represent and share knowledge about a domain by modelling the things in that domain and the relationships between those things. These relationships describe the properties of those things; in essence, what it is to be one of those things in the domain being modelled. This section presents the commonly used biomedical ontologies and terminologies in the clinical medicine and cancer domain. These ontological resources have various usages within the domain of biomedicine in general and in oncology and oncology-related biology in particular. The widest service which they provide is that of a good dictionary, where different classes, terms, entities are given unique identification codes and can be used in a way that they are univocal. Arguably this is the simplest service that ontologies can provide. Ability to draw inferences, relationship among entities at various levels of granularity, existential dependence, etc. is their more advanced services. These services are used for life-science data integration, integration of Electronic Health Record data, patient status description, and drug delivery information provision in the domain of oncology. Specific features of these terminologies and ontologies make them relevant for clinical practice in oncology and for oncology-related biomedical research, and their use possibly beneficial to the TUMOR project.

References

J. B. L. Bard, S. Y. Rhee Ontologies in biology: design, applications and future challenges, Nature Reviews Genetics, Vol. 5, No. 3. (01 March 2004), pp. 213-222.

O. Bodenreider, R. Stevens Bio-ontologies: current trends and future directions. Briefings in bioinformatics, Vol. 7, No. 3. (1 September 2006), pp. 256-274. doi:10.1093/bib/bbl027

A. Ruttenberg, T. Clark, W. Bug, M. Samwald, O. Bodenreider and H. Chen et al., Advancing translational research with the semantic web, BMC Bioinformatics 8 (Suppl. 3) (2007), p. S2.

National Cancer Institute Thesaurus

Maintained by: National Cancer Institute

Latest version: v11.02d

Availability: http://ncit.nci.nih.gov/

Description

The National Cancer Institute (NCI) Thesaurus is an ontology-like vocabulary that includes broad coverage of the cancer domain, including cancer related diseases, findings and abnormalities; anatomy; agents, drugs and chemicals; genes and gene products and so on. In certain areas, like cancer diseases and combination chemotherapies, it provides the most granular and consistent terminology available. It combines terminology from numerous cancer research related domains, and provides a way to integrate or link these kinds of information together through semantic relationships. The thesaurus currently contains over 34,000 concepts, structured into 20 taxonomic trees. NCI Thesaurus is available for free use within the European Union within the terms of its license.

Within the context of TUMOR, the NCI Thesaurus could be used as a source for standard medical terminology for annotating models with biological entities. The thesaurus is widely used internationally and is commonly used as a source for vocabulary used in biological and medical ontologies.

References

S. De Coronado, M.W. Haber, N. Sioutos, M.S. Tuttle and L.W. Wright, NCI Thesaurus: using science-based terminology to integrate cancer research results, Medinfo 2004 (2004), pp. 33–37

Sioutos N, de Coronado S, Haber MW, Hartel FW, Shaiu WL, Wright LW: NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information. Journal of biomedical informatics 2007, 40:30-43.

Foundational Model of Anatomy (FMA)

Maintained by: University of Washington

Latest version: -

Availability: http://sig.biostr.washington.edu/projects/fm/

Description

The FMA is developed by the Structural Informatics Group, University of Washington and it's concerned with the representation of classes and relationships necessary for the symbolic representation of the structure of the human body in a form that is understandable to humans and is also navigable by computerised systems. Specifically, the FMA is a domain ontology that represents a coherent body of explicit declarative knowledge about human anatomy. FMA has four interrelated components:

- Anatomy taxonomy: classifies anatomical entities according to the characteristics they share and by which they can be distinguished from one another.
- Anatomical Structural Abstraction: specifies the part-whole and spatial relationships that exist between the entities represented in the taxonomy
- Anatomical Transformation Abstraction: specifies the morphological transformation of the entities represented in the taxonomy during prenatal development and the postnatal life cycle
- Metaknowledge: specifies the principles, rules and definitions according to which classes and relationships in the other three components of FMA are represented.

FMA contains approximately 72,000 classes, over 115,000 terms and over 2.1 million relationship instances from 168 relationship types.

FMA is very useful for representing anatomical entities in relevance to oncology. These include carcinoma staging, locations for radiotherapy and surgery, access routes for various procedures, locations for drug actions, and so on. The robust formalism allows derivation of inferences, especially for staging of carcinomas.

Integrating the FMA into TUMOR could facilitate validation in model coupling, in particular where biological interfaces are linked and biological entities might interact.

References

Rosse C, Mejino JVL.. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. J Biomed. Inform. 36:478-500, 2003

International Classification of Diseases (ICD)

Maintained by: World Health Organisation

Latest version: ICD-10 2nd Edition

Availability: http://www.who.int/classifications/icd/

Description

ICD is designed to promote international comparability in the collection, processing, classification, and presentation of diagnostics in health epidemiology, health management and mortality statistics. These include the analysis of the general health situation of population groups and monitoring of the incidence and prevalence of diseases and other health problems in relation to other variables such as the characteristics and circumstances of the individuals affected. The top classes consist mainly of diseases classified according to the body system, though neoplasms, infectious diseases and injuries and poisonings have their own axes.

To a large extent, ICD provides a disease classification on the basis of anatomy. Although not all the diseases within ICD are classified according to anatomy, the neoplasms are more or less classified within the anatomical partition. Thus, an ontology of carcinomas that follows the anatomical partition for classification of neoplasms and related diseases can use portions of ICD more easily than other disease classifications. However there are issues of misclassifications within ICD and also terms, which do not represent a real disease. With certain modifications, integration of ICD with FMA related anatomy is possible in a way that inferences can be drawn on the basis of the anatomy ontology of FMA.

ICD could be used within TUMOR as a standard dictionary for describing cancers tackled by models uploaded to the EC repository.

Systematized Nomenclature of Medicine – Clinical Terms

Maintained by: International Health Terminology Standards Development Organisation

Latest version: -

Availability: http://www.ihtsdo.org/snomed-ct/

Description

SONMED CT is a generic healthcare terminology together with various relations between it's over 300,000 concepts. There are about a million descriptions of those concepts and about a million semantic links between them. The SONMED CT core content consists of:

- Concepts Table
- Descriptions Table
- Relationship Table
- History Table
- ICD Mapping

The main top classes consist of Clinical Finding, Procedure, Observable Entity, Body Structure, Organism, Substance, Pharmaceutical/Biologic Product, Specimen and Events. SONMED CT classifies attributes according to the top classes. While some attributes are used across many top classes, there are many that are characteristically used within a single top class. For example, Clinical Finding top class is associated with attributes like Severity, Onset, Course, Episodicity, Stage and so on. Similar, for Procedure, the attributes include Procedure Site, Procedure Device, Procedure Morphology, Access and so on. SONMED CT is available under license for the countries within the European Union.

SNOMED CT has been adopted internationally and maps to various other standard medical terminology dictionaries. Its potential use within TUMOR as a source of standard terminology could lead to the EC repository's markup and Web services more interoperable with other existing systems that have adopted the same vocabulary.

References

Stearns MQ, Price C, Spackman KA, Wang AY. SNOMED clinical terms: overview of the development process and project status. Proc AMIA Symp 2001:662–6

ACGT Master Ontology

Maintained by: Institute for Formal Ontology and Medical Information Science (IFOMIS)

Latest version: v1.0

Availability: http://www.ifomis.org/wiki/ACGT_MO

Description

The ACGT Master Ontology (ACGT-MO) is the result of the semantic (ontology-based) data integration in the ACGT European project. The goal of the EU co-funded project "Advancing Clinico-genomic Trials on Cancer – Open Grid Services for Improving Medical Knowledge Discovery" (ACGT) was to develop an ontology-driven, semantic grid services infrastructure that will enable efficient execution of discovery-driven scientific workflows in the context of multi-centric, post-genomic clinical trials.

The ACGT-MO is implemented in OWL-DL, the description-logics based subtype of the Web Ontology Language (OWL) and can be freely downloaded from http://www.ifomis.org/acgt. It is re-using Basic Formal Ontology (BFO, http://www.ifomis.org/bfo) as upper level and the OBO Relation Ontology (http://www.obofoundry.org/ro/) (see Fig. 1).

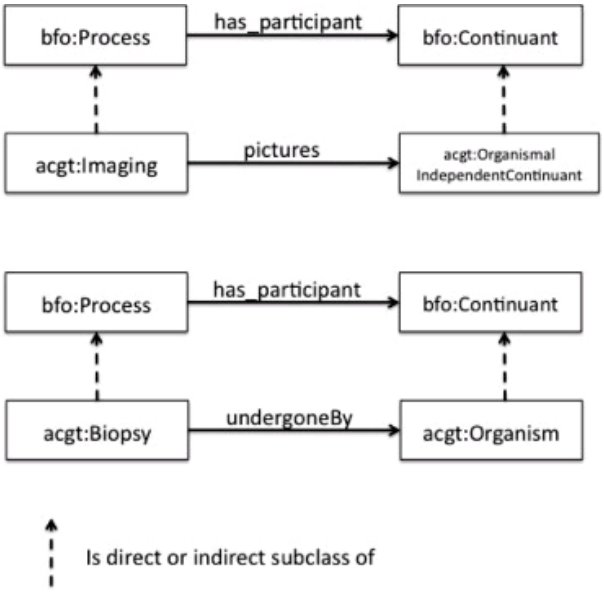

Fig. 1 Relations between ACGT-specific classes and their superclasses from BFO

The ACGT-MO developers set out to comprehensively represent the domain of cancer research and management, with special emphasis on mammary carcinoma ("breast cancer"), Wilms' tumour (nephroblastoma) and rhabdoid tumour and its development was guided and reviewed by researchers from two pre-existing clinical trials on Breast Cancer and Nephroblastoma. As a result of this development process, it not as a comprehensive domain ontology, but rather as an application ontology tailored to the needs of the ACGT software system. The ACGT-MO therefore is an application ontology and its main role, in the context of the translational medicine research framework within which it is developed and applied, is to support data integration across the borders of countries and disciplines, languages and professional terminologies; as well as integration of newly gathered data with data already stored. As a result, the ACGT-MO is heavily used in the context of the ACGT Semantic Mediation Process.

The TUMOR repository aims to store cancer models taken from the ACGT and ContraCancrum projects, and to reuse some portions of ACGT software, the reuse of the ACGT-MO could be beneficial for the TUMOR project.

References

M. Brochhausen, A. D. Spear, C. Cocos, G. Weiler, L. Martin, A. Anguita, et al., The ACGT Master Ontology and its applications - Towards an ontology-driven cancer research and management system, Journal of Biomedical Informatics, Volume 44, Issue 1, Ontologies for Clinical and Translational Research, February 2011, Pages 8-25, ISSN 1532-0464, DOI: 10.1016/j.jbi.2010.04.008.

6 TumorML: A Markup Language for Computational Cancer Modelling

To address the specific domain of cancer modelling, we propose the development of a markup language, ***TumorML***, to describe computational cancer models within the TUMOR project. The motivation for such a markup language is two-fold:

1. To describe the implementation of these cancer models in an abstract manner that is not tied to any particular programming notation
2. To be able to couple our models to address the transatlantic scenarios such as those described in D2.1.

The challenges posed in developing TumorML include:

- Formalising cancer terminology
- Linking biological entities with computational and mathematical elements of models
- Incorporating features to allow for curating models in online repositories.

Paired with ontologies of how entities of cancer biology are related and interact, and by using standard terminology dictionaries, for example the ACGT Master Ontology and the NCI Thesaurus, we will be able to package models with metadata that is standardised throughout the TUMOR repository. This would assist in modellers developing models composed of other models, irrespective of scale (e.g. molecular/microscopic to macroscopic scales) and source (e.g. models from CViT DMR or the EC TUMOR repository). Linking different models together will produce more accurate compound models, particularly when considering models operating from different scales in our bottom-up and top-down approaches described in D2.1.

Conceptual Design

Conceptually, the design of TumorML will take a similar approach to that of CellML and ISML in how models are structured to allow modularisation and connectivity between component models. In the case of TumorML however, we propose to reuse the JSDL vocabulary since we initially target models published as pre-complied model binaries, or source-code implementations that can be compiled on-the-fly (since we have determined that MathML-based descriptions of cancer models may not be sufficient within TUMOR). This means there are two key levels of abstraction when publishing a model:

1. A computational description of the model implementation
2. The biological description of the model function described by the aforementioned implementation.

When analysing the kinds of models provided by ACGT, ContraCancrum, and from CViT's DMR, we determined that an essential part of enabling model execution and workflow composition is the specification of the computational requirements to run the models. JSDL provides markup that can describe the hardware and software requirements of a binary or source file. It also allows the specification of standard inputs, outputs, data staging, and execution parameters.

Once the input and output parameters are defined at a computational level, these could be mapped to entities from cancer biology. This might allow us to perform some type checking and units conversion where possible when presented with input data, and additionally semantic checks of the biological parameters to ensure scientific correctness when connecting multiple models together. Directly connecting the computational parameters between models would not serve to validate any semantic connectivity, as raw parameters

do not have any semantic metadata attached to them by default. We term a model that is not made up of component parts a ‘Simple Model Description’ while a model that is composed of multiple model descriptions a ‘Complex Model Description’.

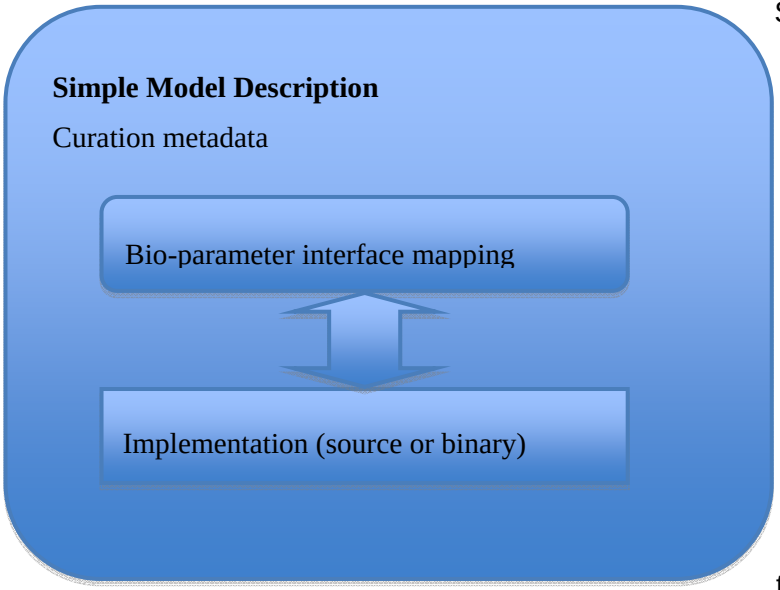

Figure 1 A 'Simple' Model Description

Simple Model Descriptions (Figure 1) are used as the initial step in wrapping up a computational cancer model. It refers to a single implementation, uploaded to the repository as a binary or as source code. An interface mapping bridges the computational interface (as command-line parameter lists or input files) with standardised biological entities (i.e. the bio-parameter interface). Models are also curated with standard metadata to enable efficient search and management of models. By making the interface to the models domain-specific, researchers and clinicians will more easily understand how to run the models and how to couple them with other models where necessary.

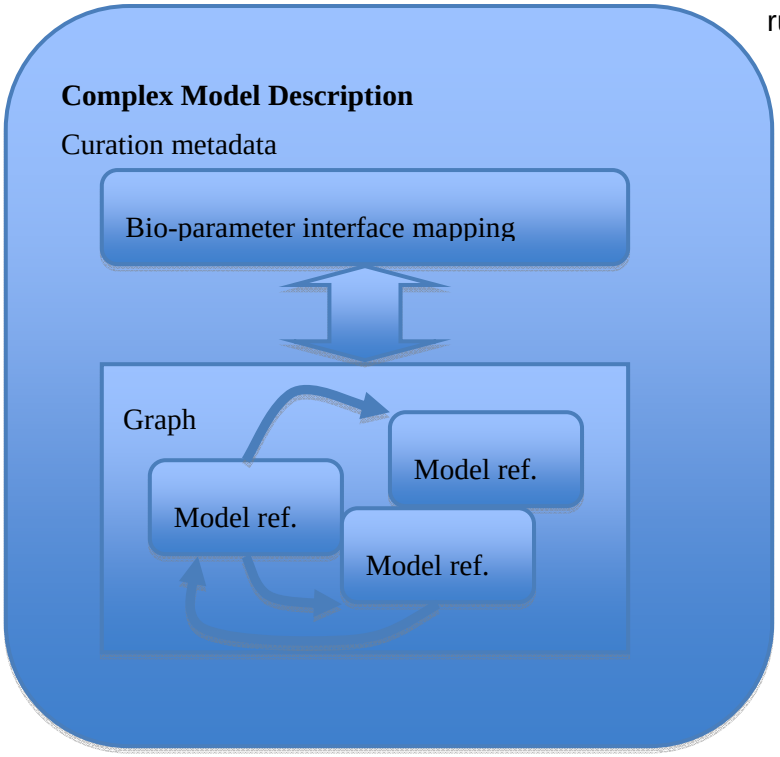

Figure 2 A 'Complex' Model Description

Complex Model Descriptions (Figure 2) provide similar functionality to Simple Model Descriptions in that they are curatable with the same metadata, and also provide a bio-parameter interface. The main difference is that as they describe a compound model, a single implementation is not referenced. Rather, a graph of references to other models describes the internal functionality of the complex model. These references may refer to any other kinds of Simple or Complex Model descriptions, and the edges of the graphs connect through each referenced model’s bio-parameter interface. The interface of the model at hand is then composed of the remaining unconnected interface components reflecting what inputs are needed and what outputs the compound model writes.

Markup for Curating Cancer Models

Like SBML, CellML, FieldML, and ISML, TumorML will utilise existing metadata vocabularies such as Dublin Core for document curation, MathML for providing validated mathematical content where possible, and where existing vocabularies do not exist we will specify our own cancer-specific metadata descriptions. Clinically oriented vocabularies and ontologies could also be integrated to assist in the management of clinical trials of TumorML models.

Work has begun in developing the database schemas for the TUMOR model repository under WP3, and is described in deliverable D3.1.1.

Through initial discussions in the first year of the TUMOR project, a basic model description was proposed that could capture domain specific elements of cancer models when searching for models in a digital repository. The CViT DMR's current approach is to provide the model publisher with free text fields describing Hypothesis, Description, Conclusion, etc. This has the advantage of being easier to enter, but may make it more difficult for the end-user to locate relevant information (or a computer program to utilize the information). The proposed metadata for TUMOR models are summarised as follows and would be in addition to Dublin Core elements. Figure 3 shows how the curation metadata is attached to the model description.

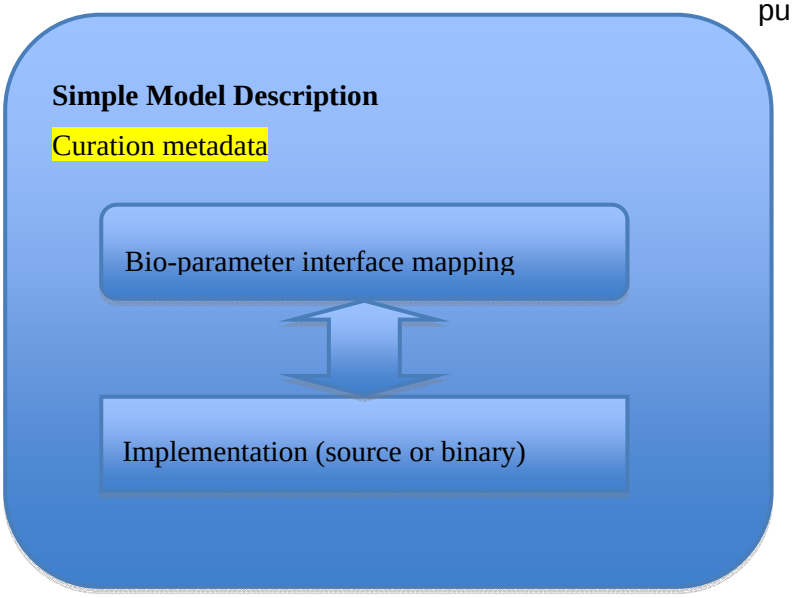

Figure 3 Conceptual view of a simple model description, highlighting the curation metadata

Model structural details: The basic structural details of the models include:

- Math type: Can be “discrete”, “continuous” or “hybrid”.
- Biocomplexity direction: Can be “Top-down”, “Bottom-up” or “Middle-out”. Can be used to bind the models of the EU to the models of the CViT repository
- Biological scales: "Atomic", "Molecular", "Cellular", "Tissue", "Organ", and "Population"
- Type of cancer: The type of cancer described by the model (breast cancer, lung cancer, etc.)
- Tumour details: Details about the tumour represented in the specific model. Described in the “Tumour details” sub-section
- Treatment details: Details about the treatment bind with the specific model. Described in the “Treatment” subsection. Various treatments can be used with each model
- Computational model details: Details about the computational parameters of the model. Described in the “Computational details” sub-section.

Tumour details: The details of the tumour described by the model. Includes the following elements:

- Materialization: Can be “solid” or “liquid”
- Homogeneousness status: Can be “macroscopically homogeneous” or “macroscopically non-homogeneous”
- Initialization: Can be “imageable” or “non-imageable tumor”
- Free-growth included: Can be “yes” or “no”

Treatment: The details of the treatment that has been used with the model. Multiple treatment schemas can be used with each model. Includes the following elements:

- Simulation duration: The duration of the whole treatment simulation. May include multiple chemo and/or radio cycles
- Treatment type: The type of treatment. Can be chemotherapy or radiotherapy
- Drug(s): The drug(s) used
- Cycle schedule duration: The duration of the treatment cycle
- Chemo cycle details: Details regarding the chemotherapy used. Described in detail in the “Chemo Details” sub-section.

Radio cycle details: Details regarding the radiotherapy used. Described in detail in the “Radio Details” sub-section.

Chemo Details: The details of the chemotherapy used with a specific treatment. May include multiple chemo cycles. For each chemo cycle, the following elements are included:

- Day: The day(s) of the drug(s) administrations
- Dose: The drug dose given
- Drug(s): The drug(s) used.

Radio Details: The details of the radiotherapy used with a specific treatment. May include multiple radio cycles. For each radio cycle, the following elements are included:

- Day: The days of radiation administration
- Hour: The hours of radiation administration
- Total Dose and Fraction Dose used.

At this early stage in the markup design, these proposed metadata details would be used to get a first prototype of the TUMOR digital repository functional. As many of these fields refer to specific entities in biology, we believe that utilising a standard dictionary of terms, such as those described in the section on state-of-the-art ontologies, would allow greater interoperability with outside repositories by being able to map terminology between different systems. In addition to this, where standard units are used to describe certain biological entities or properties, we will be able to check and automatically convert units where necessary.

Markup for Interfacing with Models

As a first step, we will develop metadata for curation and for describing the public interfaces with existing models that have been developed and published as source code and

executable files. This will allow us to investigate how to fuse models of different scales together through their exposed parametric inputs and outputs; an initial 'black box' approach to computational model execution and coupling. Parametric interfaces will be described using markup that will facilitate the specification of the underlying computational requirements for executing computational models. These computational interfaces could then be mapped to biological terminology ultimately providing a way to more easily validate the cancer biology through correct semantic matching, but also to provide a means to enforce type and units checking where heterogeneities in model descriptions exist.

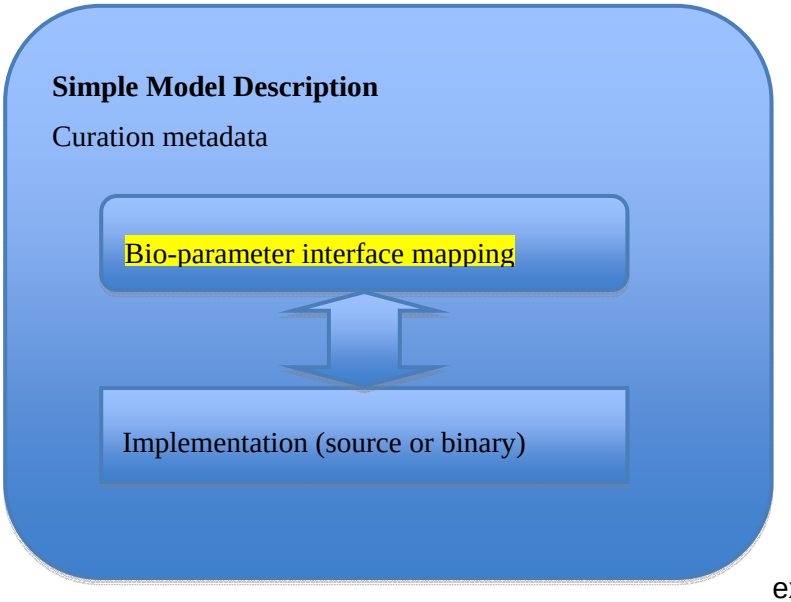

Figure 4 A simple model with the parametric interface highlighted

Parametric interfaces will be described using JSDL, which will also facilitate the specification of the underlying computational requirements for executing models. While the cancer modelling community adopts TumorML for publishing existing models, we will work with modellers to develop the next level of more detailed abstraction in the structural, mathematical, and algorithmic descriptions of the inner workings of models. Significant effort might be required to port existing models to TumorML, so by providing multiple levels of abstractive notation in our markup we can wrap existing models in early versions of TumorML as well as develop new models with an evolving markup specification.

Markup for Connecting Model Interfaces

As described previously, connecting models together will be paramount to investigate combining approaches and developing more accurate models through such composition. Where models are published as computational applications, as we envisage within the scope of the TUMOR project and as provided by the TUMOR model repository, linking models together essentially equates to workflow composition. Building workflows of computational applications is not novel and has been demonstrated by a number of well-known workflow systems such as the UK myGrid project's Taverna software, and the US Kepler project. As described in the markup review, Taverna utilises XScufl for workflow descriptions; however, we will also investigate the use of BPEL as the choice of markup for workflows. This is because the ACGT project developed a workflow-authoring tool that could potentially be used as the basis for a model composition in TUMOR within the workflow environment to be developed in WP5.

The basic idea behind any workflow composition is to build a graph of dependencies between computational modules. As we plan to investigate mapping JSDL parametric interfaces to biological entities, a model composition within the TUMOR environment would link these bio-computational interfaces together. We envisage that when the computational parameters are linked between models during the composition process, semantic validation, including type and units checking, will be carried out where the user will be given warnings on-the-fly. This would allow users to create compound models with immediate feedback, and thus reduce the likelihood of errors and execution problems that would arise at run-time.

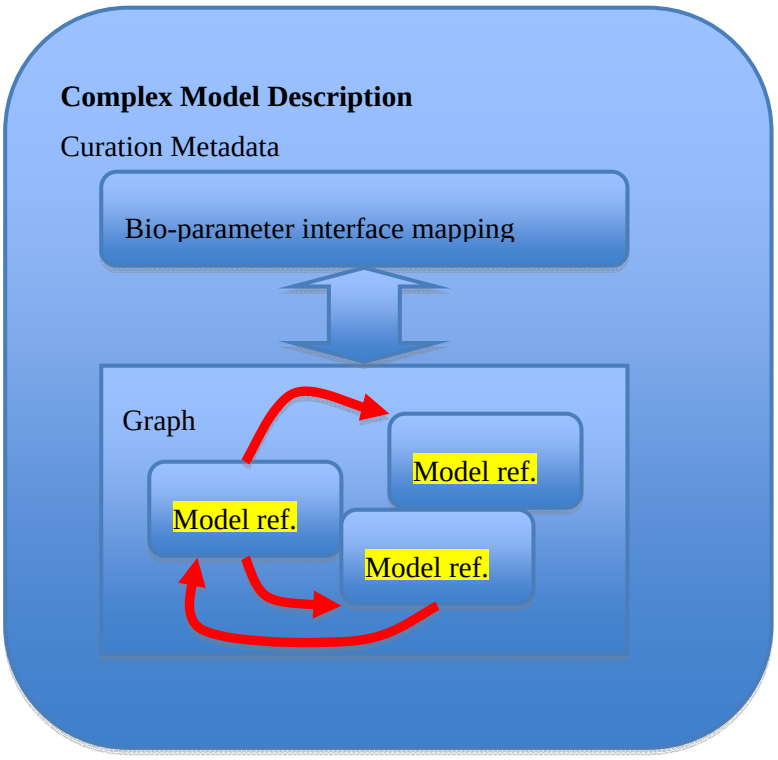

Figure 5 Complex model highlighting the details of the internal graph structure. Model references (yellow) are linked together through their model interfaces as a graph-like structure (red)

Deliverable XML Schemas, Stylesheets, and Support Tools

Apart from a high-level schema design and specification for TumorML, during the next stage of WP4 we aim to develop a set of XML software applications and support tools. The deliverables associated with T4.2 will include:

- An XML schema to allow TumorML documents to be validated against the markup specification
- XSLT stylesheets to extract metadata relating to specific vocabularies (e.g. curation metadata, model interfaces, computational requirements etc.)
- Programming APIs to assist use of TumorML in software, in particular with the TUMOR model repository (PHP, JavaScript)
- A graphical authoring tool for users not familiar with programming languages or raw XML authoring
- Documentation, including technical specifications, tutorials, examples.

We expect to deliver all of the aforementioned by PM24, where they will be published on the TUMOR website for public access.

7 Conclusion

This deliverable document reviews the state-of-the-art biological modelling languages that could be considered appropriate for cancer modelling within the TUMOR project. We subsequently examined a range of metadata vocabularies that have the potential for integration into a new markup language for cancer modelling. We also review the current state-of-the-art medical ontologies and are available that could be used to solve interoperability issues that might arise in the future by providing standardised medical and domain-specific vocabularies or through ontology mapping. Finally, we outline our proposed work on TumorML, a domain-specific markup language for in silico cancer modelling.

Our review of the currently available modelling languages (SBML, CellML, FieldML and ISML) illustrates the need for both a domain-specific approach to computational biomodelling and for one that is directed at the cancer-modelling domain. SBML is developed for the realm of systems biology – a broad ranging domain, but nonetheless a specific kind of biological modelling at the molecular scale, and hence applicable only to a subset of the models considered in TUMOR. CellML and ISML take a more generic approach and are not specifically constrained to a particular domain, although CellML was developed primarily to describe biological cell function. Both are however limited in the kinds of models they can represent. FieldML is still in development and is not yet widely adopted, and for the most part models physiological structures and their function. None of these modelling languages satisfies the needs and diversity found in cancer modelling. One other feature that is prevalent throughout is that these state-of-the-art modelling languages are designed to mainly simulate through pure mathematical description. Each language being based on MathML restricts their expressivity in modelling; especially where an in silico approach needs more algorithmic descriptions, for example in agent-based models.

We go on to review a set of markup vocabularies that could then be used to compose a new markup language tailored specifically to TUMOR's needs. Although there are many markup languages that could be incorporated into a TUMOR-specific language, we limit the scope of the proposed new cancer modelling language, *TumorML*, to three key functions: curating cancer models, computationally interfacing with cancer models, and connecting cancer models together. We propose to use Dublin Core along with cancer domain-specific metadata for model curation. We propose to use JSDL as the primary interface with published model implementations (as executable files) and map the computational interfaces with cancer domain-specific terminology and ontologies to aid in type and units checking. Finally, we propose to use a workflow markup language such as BPEL to enable the fusion of models, connecting their interfaces to form graph-like structures representing compound multiscale cancer models.

Appendix I - Abbreviations and acronyms

ACGT – Advancing Clinico Genomic Trials on Cancer

BPEL – Business Process Execution Language

caBIG – cancer Biomedical Informatics Grid

CDA – Clinical Document Architecture

ContraCancrum - Clinically Oriented Translational Cancer Multilevel Modelling

CViT – Center for the Development of a Virtual Tumor

DAE – Differential Algebraic Equation

DMR – Digital Model Repository

DTD – Data Type Definition

EC – European Commission

FEM – Finite Element Method

FOAF – Friend of a Friend

HL7 – Health Level 7

IETF – Internet Engineering Task Force

ISML – In Silico Modelling Language

JSDL – Job Submission Description Language

MathML – Mathematical Markup Language

MGH – Massachusetts General Hospital

MML – Medical Markup Language

NCI – National Cancer Institute

NIH – National Institute of Health

ODE – Ordinary Differential Equation

PDE – Partial Differential Equation

RDF – Resource Description Framework

Scufl – Simple Conceptual Unified Flow Language

SED-ML – Simulation Experiment Description Modelling Language

SBML – Systems Biology Modelling Language

SNOMED – Systematized Nomenclature of Medicine

TUMOR – Transatlantic Tumor Model Repositories

vCard – Virtual Card

VPH – Virtual Physiological Human

W3C – World Wide Web Consortium

WSDL – Web Services Description Language

XML – Extensible Markup Language

XScufl – XML Simple Conceptual Unified Flow Language

XSLT – Extensible Stylesheet Language Transformations