TumorML: Concept and Requirements of an *In Silico* Cancer Modelling Markup Language

David Johnson, Jonathan Cooper and Steve McKeever

***Abstract*— This paper describes the initial groundwork carried out as part of the European Commission funded Transatlantic Tumor Model Repositories project, to develop a new markup language for computational cancer modelling, *TumorML*. In this paper we describe the motivations for such a language, arguing that current state-of-the-art biomodelling languages are not suited to the cancer modelling domain. We go on to describe the work that needs to be done to develop TumorML, the conceptual design, and a description of what existing markup languages will be used to compose the language specification.**

I. Introduction

The European Commission funded Transatlantic Tumor Model Repositories project (TUMOR) aims to develop a European clinically oriented digital model repository for cancer models. The repository will store models provided by other European VPH projects, in particular the Advancing Clinico Genomic Trials on Cancer (ACGT) [1] and the Clinically Oriented Translational Cancer Multilevel Modelling (ContraCancrum) [2] projects. One of the aims of TUMOR is to design the digital repository to interoperate with its counterpart repository in the United States, developed by the Center for the Development of a Virtual Tumor project (CViT) [3], led by the Massachusetts General Hospital in Boston. A major task within TUMOR, and the topic of this paper, is to develop a markup language for the cancer modelling domain.

A. Motivation

We have previously discussed the motivations for modelling markup [4]. Firstly we want to describe the implementation of these cancer models in an abstract manner that is not tied to any particular programming notation, and secondly we want to be able to couple our models. To our knowledge, this kind of work has not been undertaken in the context of cancer modelling, which has led to a diverse landscape of cancer models, most of which cannot easily be used by different research groups. The development of a markup language for cancer models will enable the provision of two features.

First, by providing an expressive metadata vocabulary researchers will be able to curate their models appropriately and publish them to an audience of research peers and clinicians wishing to trial published models. To demonstrate this, models taken from ACGT and ContraCancrum will be published directly to the European repository by wrapping the computational components (as source code and/or executable binaries) in the newly developed markup language. Through Web services developed as part of the European repository's infrastructure, CViT models will be imported into the European repository where the US model metadata will be appropriately translated for storage by TUMOR services.

Second, 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 enable model coupling. To demonstrate this feature, an 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 CViT 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 (www.e-cell.org), the CellML (models.cellml.org) and FieldML (models.fieldml.org) repositories, BioModels (www.ebi.ac.uk/biomodels/) and the CViT Digital Model Repository (www.cvit.org). However most of the aforementioned repositories store models covering a wide range of biological phenomenon. TUMOR will provide a dedicated cancer model repository for European and international researchers.

B. Related Work

Existing markup languages for modelling biological phenomena include the Systems Biology Markup Language (SBML) [5], CellML [6], developed out of the cardiac modelling community, FieldML [7], a markup language primarily for modelling physiological structures and their physics, and the In Silico Markup Language (ISML) [8] developed by the Japanese Physiome project.

SBML is developed for the realm of systems biology—a broad ranging domain, but nonetheless a specific kind of

This work was supported in part by the European Commission under the Transatlantic Tumor Model Repositories project (#FP7-ICT-2009.5.4-247754). The authors would also like to thank the participating institutions of the project consortium and its members.

David Johnson is with the Department of Computer Science, University of Oxford, Wolfson Building, Oxford, OX1 3QD, United Kingdom. david.johnson@cs.ox.ac.uk

Jonathan Cooper is with the Department of Computer Science, University of Oxford, Wolfson Building, Oxford, OX1 3QD, United Kingdom. jonathan.cooper@cs.ox.ac.uk

Steve McKeever is with the Department of Computer Science, University of Oxford, Wolfson Building, Oxford, OX1 3QD, United Kingdom. steve.mckeever@cs.ox.ac.uk

modelling for molecular scale processes, and hence applicable only to a subset of the models considered in TUMOR (i.e. ACGT and ContraCancrum provide macroscopic 'top-down' models, while CViT will provide molecular/microscopic 'bottom-up' models). 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.

II. The Tumor Markup Language

To address the specific domain of cancer modelling, we propose the development of a markup language, *TumorML*, to describe computational cancer models within TUMOR. The motivation for such a markup language is two-fold: To describe the implementation of these cancer models in an abstract manner that is not tied to any particular programming notation, and to be able to couple our models to address transatlantic scenarios such as fusing one taken from CViT with one from ACGT or ContraCancrum.

The challenges posed in developing TumorML include formalising cancer terminology, linking biological entities with computational and mathematical elements of models, and 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 [13] and the National Cancer Institute Thesaurus [14], we may be able to package models with metadata that automates coupling different cancer models together, irrespective of scale and source. Linking different models together can produce more accurate compound models, particularly when considering models operating from different scales.

A. 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 components. In the case of TumorML however, we propose to reuse the Job Submission Description Language (JSDL) vocabulary, an open standard for describing computational job executions, since we initially target models published as pre-complied model binaries, or source-code implementations that can be compiled on-the-fly. This means there are two key levels of abstraction when publishing a model: (1) A computational description of the model implementation and (2) The biological description of the model function encoded in the aforementioned implementation.

An essential part of enabling model execution and workflow composition will require description of the computational requirements to run the models. Each of ACGT, ContraCancrum, and CViT, provide models as source code and binary executable files rather than more abstract representations of the model functionality. JSDL provides markup that can describe the hardware and software requirements of a binary executable. 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 will be mapped to entities from cancer biology. This will allow us to perform type checking and units conversion where necessary when presented with input data, and additionally support 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 distinguish two classes of model descriptions: Simple and Complex. Simple Model Descriptions 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. a *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.

Complex Model Descriptions 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 they describe a compound model, and so 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.

Distinguishing between Simple and Complex models is important because Complex models aggregate Simple models, and as such present an interface based on the set of encapsulated Simple models. A Complex model therefore links to the metadata of its component models, while Simple models do not as they do not refer to any additional components.

B. Markup for Digital Curation

Like SBML, CellML, FieldML, and ISML, TumorML will utilise existing vocabularies such as Dublin Core for document curation, or MathML for providing validated mathematical content where possible. 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. The CViT Digital Model Repository'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 for a computer program to utilize the information). For TumorML, we aim to incorporate the Dublin Core Metadata Element Set, a metadata vocabulary for annotating generic electronic resources, along with a set of domain-specific elements describing various features of cancer models. These would include elements describing the type of cancer modelled, the type of mathematics used in the model, what scale the model operates at, and whether any treatment scheme is incorporated into the model. Listing 1 illustrates an example of using RDF and Dublin Core to annotate a module modelling an epidermal growth factor receptor (EGFR) pathway.

Listing 1. Example of potential metadata using Dublin Core

```
 ...
<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://tumor.eu/dataservice?model_id=22">
      <dc:creator>David Johnson</dc:creator>
      <dc:title>EGFR pathway module</dc:title>
      <dc:description>An example module from the CViT team of an implementation
                      of an EGFR pathway.</dc:description>
      <dc:date>2010-09-05</dc:date>
   </rdf:Description>
</rdf:RDF>
 ...
```

At this early stage in the markup design, these proposed metadata details will be used to get a first prototype of the European 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 above, 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.

C. Markup for Interfacing with Models

We will also develop metadata 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 fuse models together through their exposed parametric inputs and outputs. This 'black box' approach to computational model execution and coupling will utilise parametric interfaces described using markup, by specifying the underlying computational requirements for executing models as metadata. The computational interfaces could then be mapped to biological terminology ultimately providing a way to validate the cancer biology more easily through correct semantic matching, and also providing a means to enforce type and units checking.

These interfaces will be described using JSDL, which will facilitate the specification of the underlying computational requirements for executing cancer models uploaded to the European repository. An example of how JSDL can be used to describe an executable model is shown in Listing 2. In this example, a model executable corresponding to running a simulation of a brain tumour (Glioblastoma multiforme) is described, with as inputs a magnetic resonance imaging (MRI) scan of the initial tumour and one parameter describing the fractional cell kill ratio applied to the simulation.

Listing 2. Example of a model executable and input argument specification

```
 ...
<jsdl:Application>
   <jsdl:POSIXApplication>
      <jsdl-posix:Executable>GlioblastomaModel</jsdl-posix:Executable>
      <jsdl-posix:Argument>inputMRI.raw</jsdl-posix:Argument>
      <jsdl-posix:Argument>CellKillRatio</jsdl-posix:Argument>
   </jsdl-posix:POSIXApplication>
</jsdl:Application>
 ...
```

We might annotate each argument to incorporate entity references and links to external publications or other information on the parameters to give them more semantic meaning. Listing 3 illustrates how one might annotate the input parameters of the model executable described in Listing 2. The `entity` annotations depicted each provide a type identifier that may be looked up against an ontology to determine the appropriate units and biological type, and a reference that is unique within the model context. Such an entity reference could be used externally to refer to specific parameters by a name or identifier rather than by the position of the argument in the list of inputs. The `linkOut` annotations illustrate how to refer to external resources, perhaps looked up as a URL (for Web links) or a DOI (for published works such as journal papers). In this example, the `mridata` parameter links out to the Wikipedia entry about MRI, and the `cellKillRatio` parameter links to a journal publication on the use of the cell kill ratio in a clinical cancer model [9].

Listing 3. Example of annotations to model arguments

```
 ...
<jsdl-posix:Argument>
   <meta>
      <entity id="MRI_1" ref="mridata"/>
      <linkOut url="http://en.wikipedia.org/wiki/Magnetic_resonance_imaging"/>
   </meta>
    inputMRI.raw
</jsdl-posix:Argument>
<jsdl-posix:Argument>
    <meta>
       <entity id="CKR_1" ref="cellKillRatio"/>
       <linkOut doi="10.1371/journal.pone.0017594"/>
    </meta>
    CellKillRatio
</jsdl-posix:Argument>
 ...
```

D. Markup for Fusing Models

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 [10] and the US Kepler project [11]. Taverna utilises the XML Simple Conceptual Unified Flow Language (XScufl) [12] for workflow descriptions; however, we will also investigate the use of Business Process Execution Language (BPEL) [15] as a possible choice of markup for workflows. This is because the ACGT project developed a workflow-authoring tool [16] that could potentially be used as the basis for model composition in TUMOR within the workflow environment to be developed as part of the model repository.

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, model composition within the TUMOR environment would link these bio-computational interfaces together. We envisage that the 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 edit-time feedback, and thus reduce the likelihood of errors and execution problems that would arise at run-time.

III. Conclusions and Future Work

A. Conclusions

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.

B. Future Work

Apart from a high-level schema design and specification for TumorML, during the next stage of development we aim to develop a set of XML software applications and support tools. These 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 the use of TumorML in software, in particular with the European model repository (e.g. in PHP, JavaScript).
- A graphical authoring tool for users not familiar with programming languages or raw XML authoring.
- Documentation, including technical specifications, tutorials, examples.

Progress on the development of TumorML will be published on the TUMOR website (www.tumor-project.eu) as will the markup specifications, schemas, tools, and documentation, and the authors openly welcome constructive contribution from the cancer modelling community.

IV. Acknowledgements

The authors gratefully acknowledge the contribution of the European Commission and the TUMOR project consortium.

References

[1] M. Tsiknakis et al., Semantic Grid Infrastructure Enabling Integrated Access and Analysis of Multilevel Biomedical Data in Support of Postgenomic Clinical Trials on Cancer, *IEEE Trans. Inf. Technol. Biomed.*, vol. 12, no. 2, March 2008, pp. 205–217.

[2] K. Marias et al., Contra Cancrum Website. Internet: www.contracancrum.eu [March 25, 2010].

[3] T.S. Deisboeck et al., Advancing Cancer Systems Biology: Introducing the Center for the Development of a Virtual Tumor, CViT, *Cancer Informatics*, vol. 5, March 2007, pp. 1–8.

[4] D. Johnson, et al., Markup Languages for In Silico Oncology, *in Proc. 4th Int. Adv. Res. Workshop on In Silico Oncology and Cancer Investigation (4th IARWISOCI) The ContraCancrum Workshop*, Athens, Greece, 2010, pp. 108-110.

[5] M. Hucka et al., Evolving a Lingua Franca and Associated Software Infrastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project, *Systems Biology*, vol. 1, no. 1, June 2004, pp. 41–53.

[6] C.M.Lloyd et al., CellML: its future, present and past, *Progress in Biophysics and Molecular Biology*, vol. 85, 2004, pp. 433–450.

[7] G. Richard Christie et al., FieldML: Concepts and Implementation, *Philosophical Transactions of the Royal Society A.*, vol. 367, no. 1895, May 2009, pp. 1869–1884.

[8] Y. Asai et al., Specifications of insilicoML 1.0 : A Multilevel Biophysical Model Description Language. *Journal of Physiological Sciences*, vol. 58, no. 7, December 2008, pp. 447–458.

[9] G.S. Stamatakos et al., Exploiting Clinical Trial Data Drastically Narrows the Window of Possible Solutions to the Problem of Clinical Adaptation of a Multiscale Cancer Model, *PLoS ONE*, vol. 6, no. 3, 2011.

[10] T. Oinn et al., Taverna: a tool for the composition and enactment of bioinformatics workflows, *in Bioinformatics*, vol. 20, no. 17, Jun. 2004, pp. 3045–3054.

[11] I. Altintas et al., Kepler: An Extensible System for Design and Execution of Scientific Workflows, *in Proc. 16th Conf. Scientific and Statistical Database Manage.*, Santorini, Greece, 2004, pp. 423–424.

[12] T. Oinn (2004, Apr. 7), XScufl Language Reference. Internet: Available: www.ebi.ac.uk/tmo/mygrid/XScuflSpecification.html [October 14, 2009].

[13] M. Brochhausen et al., The ACGT Master Ontology on Cancer A New Terminology Source for Oncological Practice, *in Proc. 21st IEEE Int. Symp. Computer-Based Medical Systems, 2008 (CBMS '08)*, Jyvaskyla, Finland, 2008, pp.324-329.

[14] N. Sioutos et al., NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information, *Journal of Biomedical Informatics*, no.40, 2007, pp. 30–43.

[15] T. Andrews et al., Business process execution language for web services version 1.1 (Technical report), May, 2003.

[16] S. Sfakianakis et al., Web-Based Authoring and Secure Enactment of Bioinformatics Workflows, *in Workshops at the Grid and Pervasive Computing Conference, 2009 (GPC '09)*, Geneva, Switzerland, 2009, pp. 88–95.