2014 IEEE 10th International Conference on eScience

Ontologies for Describing the Context of Scientific Experiment Processes

Rudolf Mayer, Tomasz Miksa
SBA Research
Vienna, Austria

Andreas Rauber
Vienna University of Technology & SBA Research
Vienna, Austria

***Abstract*—The re-usability and repeatability of e-Science experiments is widely understood as a requirement of validating and reusing previous work in data-intensive domains. Experiments are, however, often complex chains of processing, involving a number of data sources, computing infrastructure, software tools, or external and third-party services, rendering repeatability a challenging task. Another important aspect of many experiments is in the social and organisational dimension – very often, knowledge on how experiments are performed is tacit and remains with the researcher, and the collaborative and distributed aspects especially of larger collaborative experiments adds to this challenge. Therefore, a number of approaches have tackled this issue from various angles – initiatives for data sharing, code versioning and publishing as open source, the use of workflow engines to formalise the steps taken in an experiment, to ways to describe the complex environment an experiment is executed in, e.g. via Research Objects. In this paper, we present a model that has a specific focus on the technical infrastructure that is the basis of the research experiment. We demonstrate how this model can be applied to describe e-Science experiments, and align and compare it to Research Objects.**

I. Introduction

The repeatability of data and computational intensive experiments is a key aspect in making their results more trustworthy, and thus also of higher relevance to other researchers in the community. Repeatability is however difficult to achieve, due to the complexity of many of the research investigations undertaken. Often, these experiments involve several researchers from different organisations, working with a multitude of data sets, on distributed computing environments, composed of a multitude of different hardware and software components. Thus, ways to formalise the execution of these experiments are required, and workflows have been proposed as one means to orchestrate and describe the chains of data processing in e-Science experiments. Furthermore, models such as Research Objects have been proposed as a means to describe the interlinking between data resources and processing steps, and the documents that are an output of the research process, such as reports and publications. This allows to not only formalise the execution orchestration, but to also attribute how multiple experiments are connected to each other, and on which resources the scientific analysis is based on. All these are a step towards repeatability.

However, there is still relatively little efforts in towards also preserving the execution environments scientific research processes are embedded in. As such, many models lack in expressiveness in regards to describing the hardware and software setup, and the interdependencies between those components. Todays data-intensive experiments do, however, often depend on a number of services and aspects of the process beyond the control of the workflow system, or even the researcher himself, as reported e.g. in [1].

In this paper, we therefore present the TIMBUS Context Model, a model that has similar aims as e.g. Research Objects, but has a specific focus as well on the technical infrastructure that is the basis of the research experiment. Our model originated from the domain of the digital preservation of processes, which aims at enabling redeployment (re-enactment) of a process even when the technical environment has changed. We then provide a in-depth comparison between these two models, and show how they can be aligned with each other. This allows for each of the models to complement the other approach, and to refine some of their concepts. We also give two examples of use case applications of the model.

The remainder of this paper is organised as follows. Section II reviews related work in the area of digital preservation and models to describe scientific experiments. Section III then introduces the TIMBUS Context Model, followed by a comparison to and integration with Research Objects in Section IV. Section V describes the example application of the Context Model on two use cases, before Section VI provides conclusions and an outlook on future work.

II. Related Work

We will review related work from several different, complementary areas that influenced the model presented in this paper. *Digital Preservation*, as defined in the UNESCO Guidelines for the Preservation of the Digital Heritage [2] is the process of preserving data of digital origin. Its goal is that digital objects remain accessible, usable and interpretable in an authentic manner, opposing changing technologies or designated user communities that would derogate these goals. To this end, it uses preservation action strategies such as migration [3] and emulation [4] to especially counter technological changes. When different user communities need to work with the digital objects, provision of meta-data is often the preferred solution. This is a scenario very similar to many of the repeatability and reproducibility concerns in experimental sciences, where researchers not involved in the

978-1-4799-4287-9/14 $31.00 © 2014 IEEE
DOI 10.1109/eScience.2014.47

initial scientific process need to work with the experiment environment.

While digital preservation has been traditionally driven by cultural heritage and memory institutions [2], it is increasingly recognised that it is a problem affecting all organisations that manage information over time. As such, it also affects scientific research, where information systems and data processing play an ever-increasingly role, notably in e-Science experiments. The DCC Curation Life Cycle Model [5] elongates the traditional scope of preservation to include curation. The *Curation* phase might involve the creation of new information or the access and reuse of already existing information and its appraisal and selection. The *Preservation* phase involves the application of preservation actions, and the storing of that information. During the two phases, activities such as preservation planning take place in order to keep descriptive metadata and representation information up to date. The SHAMAN Information Life Cycle [6] suggests phases that aim at the capturing of the context of production of the object, and the preparation of the retrieved digital objects so that its information contents can be used. The Work by the CASPAR project on Preservation Networks [7] focuses on the capturing of the dependencies of complex digital objects through the usage of entity-relationship-like models, although business and organisational aspects are left out of it. This approach is similar to ontology-based representations of the process environment.

Workflow-Centric Research Objects [8] (ROs) are a means to aggregate or bundle resources used in a scientific investigation, such as a workflow, provenance from results of its execution, and other digital resources such as publications, data-sets. In addition, annotations are used to further describe these digital objects. The model of Research Objects is in the form of an OWL ontology, and incorporates several existing ontologies. At its core, the Research Object model extends the *Object Exchange and Reuse* model (ORE) [9][1] to formalise the aggregation of digital resources. Annotations are realised by using the Annotation Ontology (AO) [10], which allows e.g. for comment and tag-style textual annotations. Specifying the structure of an abstract workflow is enabled by the *wfdesc* ontology. Finally, the provenance of a specific execution of a workflow is described using the *wfprov* ontology. Research objects have also been presented as a means to preserve scientific processes [11], proposing archiving and autonomous curation solutions that would monitor the decay of workflows.

Enterprise architecture (EA) takes a holistic point of view on enterprise analysis, design and planning. It regards aspects such as business, information, process and technology, and does not address any specific domain-dependent concerns. It rather cuts across the whole organisation running the process, [12]. EA is thus a major driver when designing a holistic model of a process, including the social, legal, organisational and technical environment it is embedded in.

[1] http://www.openarchives.org/ore/1.0

III. Process Context Meta-Model

Our process context meta-model was driven by requirements to preserve and re-execute complete processes, from the business or scientific domain. Preservation of business processes is of key importance for companies for issues such as liability cases, where e.g. a company needs to prove that it executed its processes correctly, and faults did not occur because of their manufacturing. In the scientific world, experiments and their results need to be verifiable by others in the community. They need to be preserved as researchers need to be able to reproduce and build on top of earlier experiments to verify and expand on the results.

As a basis for the preservation of processes, we designed a meta-model that would capture the context this process is embedded in. This context can range from immediate and local aspects such as the software and hardware supporting the process, to aspects such as the organisation the process is executed in, the people involved, service providers, and even laws and regulations. The exact context can differ significantly depending on the domain the process stems from.

Therefore, the main guiding principle for designing the meta-model was for it to be modular and extensible to new aspects a use case might require. The meta-models must follow the principles of high-cohesion and low-coupling, which contributes to expressiveness and extensibility. It is especially important that adding new domain-specific aspects does not interfere with the concepts already present. Ontologies were identified as a suitable means to author the meta-model, and we decided to adopt the Web Ontology Language (OWL) [13], a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. The architecture of the proposed context model is thus based on the following concepts (cf. Figure III):

- A core model describing generic concepts of processes
- Extensions describing specific aspects regarding digital preservation of processes
- Extensions describing use case domain specific aspects
- Ontology Integration
- Model Transformation

The Core ontology, also referred to as Domain Independent Ontology (DIO), represents a neutral, domain-independent language that is able to represent the core concepts of the context model. It is designated domain-independent since it does not address any specific domain-dependent concerns, but rather cuts across the whole organisation running the process, in a similar manner as for example enterprise architecture frameworks model businesses[12]. In ontology engineering such an ontology is referred to as an “upper level ontology”.

An extension ontology represents a more specific language that addresses particular concerns, e.g. details on software, hardware, licenses, or file formats. Extensions might be arranged in hierarchies, where e.g. software licenses are described in more detail for both free and open source and commercial licenses, each having different sets of properties. The ontology integration described above makes use of model

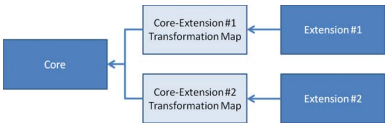

Fig. 1. Mappings between Core (generic) and extension (specific) ontologies

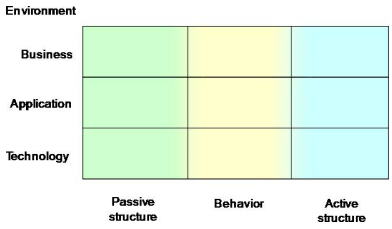

Fig. 2. The ArchiMate Framework ([16])

transformation to relate an extension to the Core, or to relate multiple extensions to each other. Model transformation entails defining a mapping strategy from a source model to a destination model ([14], [15]).

A. Core (Domain-Independent) Ontology

To ground our approach to context modelling and address the aforementioned principles, we decided to use the ArchiMate 2.0 language ([16]) as a basis for our Core ontology. ArchiMate is an international standard that covers the domain of enterprise architecture. The ArchiMate modelling language includes a minimum set of concepts and relationships and the framework includes a minimum set of layers and aspects to enable modelling of the majority of cases ([16]). Therefore, it can be considered a domain-independent language in the setting of enterprise architecture. The motivation to select the ArchiMate is that its design principles largely overlap with those of the context model.

The framework organises the modelling language concepts in a 3×3 matrix: the rows capture the enterprise layers, i.e., business, application, and technology, and the columns capture cross layer aspects, i.e., active structure, behaviour and passive structure. Figure 2 depicts this organisation of the framework. Figure 3 depicts the main concepts provided by ArchiMate, the colours of the elements corresponding to the categorisation into active structure, behaviour and passive structure. The business layer is concerned with products and (external) services, realised by the business processes of the organisation. The application layer deals with the application services, which support the business layer and are realised by software applications. The technology layer finally describes the infrastructure services offered to applications, realised by hardware and system software. Regarding aspects, the active structure contains entities capable of performing behaviour; the behaviour, contains elements defined as units of activity performed by one or more active structure elements; and the passive structure contains objects on which behaviour is performed.

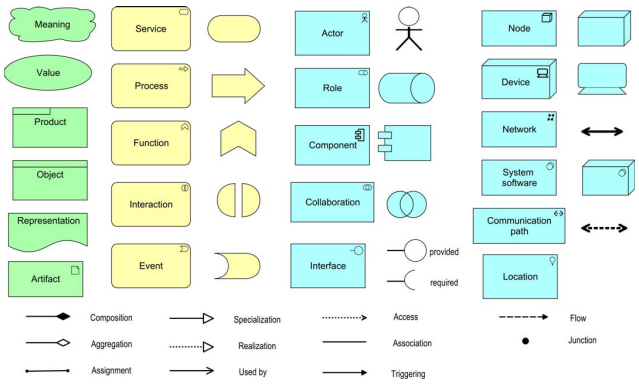

Fig. 3. The ArchiMate meta-model language concepts [16]

The possible relationships between concepts can be either inter-layer dependencies between two layers, which are usually fulfilled by the "Used By" relationship, where the lower-level layer usually provides a service which is used by elements at the higher level layer, when an element at a higher layer is realised by an element at a lower layer, or when a lower layer element is assigned to a higher layer element. Intra-layer relations normally define dependencies, e.g. between different Software components, or from Software to Hardware.

B. Extension Ontologies

The core ontology of the Context Model described above is augmented through a set of specific extension ontologies, each of which tailored to address explicit modelling concerns. Wherever possible, the extension ontologies are based on already existing languages, for which then the ontology mapping to the core ontology was provided. However in some cases, new ontologies had to be developed, based on existing vocabularies. Through the analysis of several use cases in and outside of the TIMBUS project, a number of categories of information aspects were identified, the most important of which are described below.

1) Software Dependencies: This category involves aspects an dependencies between different types of software, including information on which versions are compatible or conflicting with each other. It is for example important to know that, and which version of, a Java Virtual Machine is required to run a specific Software, or that a certain application is required to view a digital object. This is important when considering preservation actions on specific parts of the software stack utilised in the process, where also the question on what alternative software application are equivalent might be of importance. Technical dependencies on software and operating systems can be captured and described via the Common Upgradeability Description Format (CUDF) [17]. CUDF defines two concepts (*package* and *virtual package*), and a number of relations between these – among others *depends*, *recommends*, *conflicts*, and *provides*. Further, a number of data properties are provided. This allows to describe in detail the current software setup on a system, and to analyse the impact of potential changes to elements of the software stack. The ontology can

be used for different types of software, e.g. Linux packages, or Windows Dynamic Link Libraries (DLLs)

2) Data Formats: In a business or scientific process, a number of digital objects are created, modified or read. This category includes information on which data format these are described in. This type of information is the main concern of traditional digital preservation activities, and thus a tight interlinking to previous results is aimed for. Information on the format of these objects is crucial for digital preservation actions to be carried out, as e.g. migration to a different format might require changes in the rest of the process.

File formats are among the main concerns of traditional digital preservation activities, and thus it is easy to identify suitable, existing ontologies. We adopted the PREMIS Data Dictionary [18], which is also available in the form of an ontology. The data dictionary defines five types of entities: Intellectual, Object, Event, Agent, and Rights. It then defines 45 concepts belonging to these types, as well as relations and data properties. To integrate PREMIS in our meta-model, we map the *File* entity to an ArchiMate *Artifact*, and can then utilise the PREMIS elements of *Format* and *FormatRegistry* to further describe them. Also, *Storage*, *ContentLocation* and *Software*, as well as *Agents* are mapped to the core ontology. Further, we are considering mappings of *Rights* to both the core and the legal extension ontologies, which are described below.

3) Hardware: This category includes aspects related to hardware, from desktop systems, computational and storage server infrastructure, to specialised hardware optimised for certain tasks. Even though in many processes the hardware employed to host the software applications might be standard commodity hardware, its exact specifications can still influence the run-time behaviour of a process. This might be critical in certain circumstances, such as execution speed, or when specific functionalities and characteristics of the hardware such as precision limits, analog/digital conversion thresholds etc. are part of the computation. Further, certain processes might utilise certain hardware capabilities for computation, such as using graphical processing units (GPUs) for large-scale experiments in scientific processes. These types of hardware, and the software that can work upon them, are not yet as standardised and abstracted, thus an exact description is needed in many cases. The ontology described in [19], developed with the aim of describing persuasive and context-aware computing, includes as main concepts *Device* and *Resource*, which can be mapped to elements in the core ontology (*Device* and *Node*).

4) Legal: This category includes all legal requirements imposed on the processes and surrounding context. For example, this can be about whether certain elements are eligible for preservation. This includes reproduction of copyright-protected data and software, as well as intellectual-property materials, e.g. software. It is, therefore, crucial that legal aspects are considered when performing digital preservation of processes, which include software and data alike. While the domain is very extensive, we consider the following fields of specific importance.

a) License: This category includes all aspects related to licenses, and concentrates initially on software licenses. Relevant aspects are e.g. the types of licenses under which software was made available, and the clauses they contain. These license clauses then pose restrictions on what can be performed with the software. Licenses are a specialisation of legal requirements, but of high importance for digital preservation aspects, thus they are described in detail in their own model. A suitable candidate for this domain was identified in a section of "The Software Ontology"[2] (SWO), which is an ontology for describing software tools, their types, tasks, versions, provenance and associated data. One of its components is dedicated to licenses, and models two important concepts: Software licenses, and License clauses. License clauses define properties and restrictions on what can be done with the software, e.g. whether redistribution is allowed, and in what form (with or without notice), or whether there is a restriction on the number of users that can use the software. Software licenses are a composition of clauses. The ontology pre-defines a set of these, others can be added as needed. The mapping of both Clause and License to the Core ontology is to the *Constraint* concept.

b) Patents: This category contains aspects on patents, e.g. who is the owner of a specific patent, what the patent covers, or when it was granted. Patents also imply a restriction on how a software, hardware or method can be used. If this is the case, it could have implications on whether, or to what level of completeness, the preservation of the processes could be performed. Again, these are a specialisation of legal requirements considered to be of high importance, and are thus described with their own set of concept. A suitable candidate is a result of the EU-funded PATExpert project[3]. PATExpert defined a suite of ontologies that describe patent documents, covering aspects such as the structure of documents and content they provide.

5) Performance and Provenance: Information on the actual execution or enactment of a process is of importance especially for verifying repeatability of two different runs. Therefore, such information is also important for the process context model. We based this extension on the Janus model [20], which provides a semantic provenance model and was designed around the Taverna workflow model. As such, it is relatively similar in structure to the Research Objects model for provenance. Janus itself is an extension of the PROVENIR ontology, which provides an upper-level reference description languages for provenance modelling that can be extended to represent provenance in multiple domains [21]. Janus then adds concepts specific for workflows. The three top level concepts are agent, data and process. Process consists of a *workflow_spec*, which specifies the orchestration of one or more *processor_spec*, which represents a process step. Subsumed under data are the concepts of *port* and the *port_value*. The former represent the input and output data

[2]http://theswo.sourceforge.net/

[3]http://cordis.europa.eu/ist/kct/patexpert_synopsis.htm

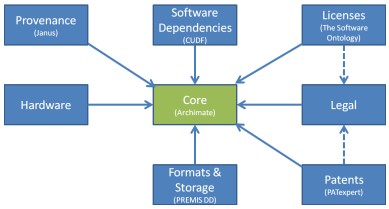

Fig. 4. Overview on extensions and their relation to the core ontology

of the process, while the latter stores the process execution data. The integration to the core ontology is towards *Business Objects* and *Process*, respectively aggregation of Processes.

It is important to note that the ontology mapping between the extensions and the core ontology is not 1:1. In most cases, there is an excess of concepts in the extension ontology, i.e. some concepts map to none of the concepts in the target model. These concepts are mostly of such nature as that they describe constructs that are mapped to the core ontology in more detail, and thus this excess of concepts is desired.

An overview on the current extensions available and their mapping to the core ontology and other extension ontologies is depicted in Figure 4. Different alternatives were analysed for creating the mappings between the extensions and the core ontology, including OWL *equivalentTo* statements, which intuitively would be a clear candidate. However, in most cases the extensions define only one possible *specialisation* of a core concept, and thus *subClass* statements were employed on the mappings. The set of extension ontologies currently available is not complete for all possible scenarios, but the extension mechanism is easily applicable to provide other domain specific ontologies that may be needed for capturing the context of other use cases. More details on the generic architecture of the Context Model, including aspects on the ontology integration, and an in-depth discussion on reasoning aspects on the ontology can be found in [22]. The ontologies are available from the TIMBUS project website[4].

IV. Comparison to and Alignment with Research Objects

In this section, we will give an analysis of how the Context Model and Research Objects complement each other, and propose an alignment that allows to integrate both models. This alignment is possible in both directions, meaning that the Context Model can be augmented by (parts of) the Research Object model, and vice-verca, (parts of) the Context Model can be used to refine some of the concepts in Research Objects. We base our comparison on the currently available version of the Wf4Ever Research Object Model Specification 1.0[5]. Schematic overviews on these ontologies are given in Figure IV. The Context Model provides more powerful concepts, and integrates especially aspects on the technology supporting the workflow or process execution. However, Research Objects are a prominent concept that has seen wide-spread discussion in the e-Science community, thus a detailed comparison and evaluation of the two models seems required.

The business layer of the Context Model Core ontology contains as most prominent concept a *Process*, which represents a single task. A set of process tasks can be aggregated to form a sub-process (sequence). The *Process* concept is thus equivalent to the *Process* in Research Objects (RO), while the RO concept of a WorkflowTemplate is equivalent to an aggregation of Processes in the Context Model. Processes produce or consume data and information, which is represented by the *Business Object*. Contrary to ROs, there is no specific distinction between *Input* and *Output* data, but the semantic differentiation is defined by the direction of the relation they are connected to with Process elements. Thus, *Object* can be considered to be a super-concept of the ROs *Input* and *Output*. In the Context Model, there is no specific concept equivalent to the *Parameter* in Research Objects. However, as parameters are a specific type of inputs, *Business Object* can be seen as a superclass and used for that purpose as well.

wfdesc:Artifact is described as "used to provide information about a class of artifacts. For example, it can be used to specify the datatype of a dataset or the structure of a document.". An similar concept in the Context Model can be found in the PREMIS extension ontology (cf. Section III-B2) in the *premis:Format*. While PREMIS is mostly oriented towards well-known data formats which can be queried from *premis:FormatRegistry*, the *premis:FormatDesignation* also allows to specify any custom format or structure a digital object might adhere to, and can thus be considered equivalent to *wfdesc:Artifact*.

The concepts of the RO workflow provenance model are in general equivalent to concepts in the Context Model Performance and Performance extension. *wfprov:WorkflowRun* is matched by *janus:workflow_run*, and *wfprov:ProcessRun* by *janus:processor_exec*. *wfdesc:Artifact*, which is defined as "a data value or item", is equivalent to an *core:Artifact*. An Artifact, a concept on the technology layer in the Context Model, is the manifestation of an abstract Business Object on the business layer. They are linked to each other via *Data Objects* on the application layer.

No equivalent mapping can currently be identified for the *wfprov:WorkflowEngine* class, which has is the agent responsible for enacting a workflow definition. The closest concept in the Context Model is the *core:SystemSoftware*, which generally describes executables and runtime environments, such as a web server, or a Java virtual machine (JVM). There is currently work in progress to more precisely define different categories for system software, e.g. in Database Systems, Compilers, or the before-mentioned web server and runtime environments for code in general (JVM, Perl interpreter, etc.). Workflow engines will likely form a dedicated category at the same level, pending final dependency analysis to Database Systems and interpreters.

In the Wf4Ever model, the *wf4ever:WebServiceProcess*,

[4]http://timbusproject.net/resources/publications/ontologies
[5]http://wf4ever.github.io/ro/2013-11-30/

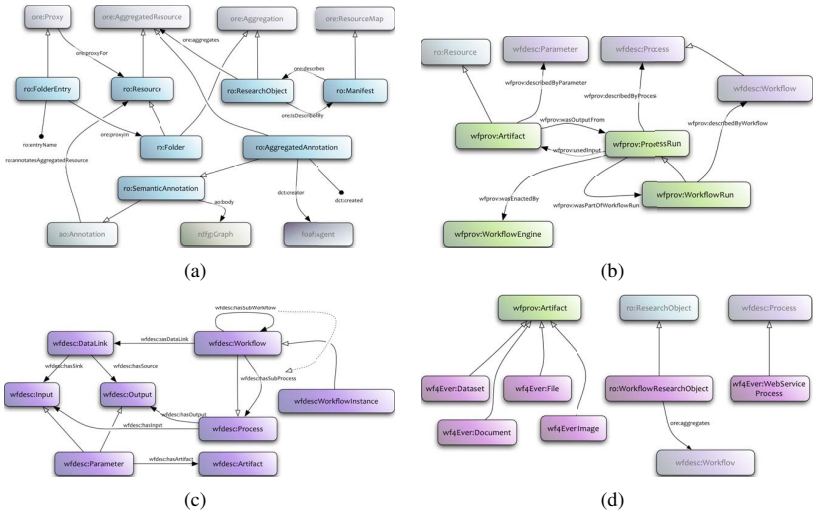

Fig. 5. Research Object Ontologies[5]: (a) Core model, (c) Workflow description model, (b) Workflow provenance model, (d) Workflow4Ever specific model

defined as a *wfdesc:Process* description, the enactment of which gives rise to a web service call", is similar to the notion of *Service* in the Context Model. However, there is no direct equivalence, only a rough equivalence to a *Service* that is usedBy a *Process*. In addition, an *Interface* object can be used to narrow the type of service to a specific interface, which allows to describe a WebService call.

wf4ever:Dataset, *wf4ever:Document*, *wf4ever:File*, and *wf4ever:Image* are all specific sub-classes of *core:Artifact*. If *core:Artifact* is further refined by a *premis:Format*, however, an equivalence to the wf4ever types can be inferred.

Regarding the Core RO ontology, the similarities between the Context Model and ROs are less obvious. Several concepts deal with aggregations and annotations thereof, such as *ro:AggregatedAnnotation*, *ro:Folder*, *ro:FolderEntry* and the *ro:ResearchObject* itself. The closest relative to this in the Context Model is enabled by the *aggregation relationship*, a structural relationship that "indicates that a concept groups a number of other concepts". Also the *composition relationship*, which "indicates that an object is composed of one or more other objects", might be applicable in some cases. A *ro:Resource* is a very broad concept in ROs, and has no direct equivalence in the Context Model; it is thus mapped as a superclass of e.g. *core:Artifact*.

An overview of identified mappings is given in Table I. In summary, it can be noted that many aspects of the Context Model and Research Objects are similar, and have been integrated in the model to tackle similar requirements. An important difference is that the Research Objects put more emphasise on different types of artifacts and how they are aggregated to form new units of information. On the other hand, the Context Model, coming from the domain of digital preservation, has a stronger focus on technical aspects, such as precise information on the software setup and dependencies, data formats interlinked with format registries, and aspects such as licenses and other legal issues. Due to the modularity of both models, each can integrate certain parts of the other to augment its expressiveness. The Context Model has more expressive power especially in aspects of hardware and software technology, file formats, and legal requirements and constraints. It is thus more suitable especially for digital preservation aspects, when the aim is to preserve the computing context the research process is executed in, to prepare for a later re-enactment. With the mapping provided above, Context Models and Research Objects can be augmented by each other, thus providing one modelling approach to focus both on technical aspects as well as the specific roles of information objects in a research process.

V. USE CASES

To show the applicability of the Context Model, we are describing two use case applications. The first use case addressed is a process of model building and analysis in the domain of sensor networks; specifically, the use case is dealing with data from large civil engineering structures such as dams. statistical, physical and mathematical models are built to predict and explain the behaviour of these structures.

The "Business" section of Figure 6 details a multiple linear regression sub-process used in dam safety monitoring. The sub-process is part of larger process, but has been isolated for demonstration purposes. Business Objects are utilised to depict both parameters to the specific steps, as well as data exchanged between them. Overall, it is composed of the following five steps:

1) Extract data: based on a set of extraction parameters, this process generates the sensor data that will be used in the multiple linear regressions (MLR) model
2) Generate regression: Based on a set of regression parameters (e.g. equation to estimate the hydrostatic effect), this process generates the regression controls that configure the parameters for the MLR model

TABLE I
EQUIVALENCE MAPPING BETWEEN CONTEXT MODEL (CM) AND RESEARCH OBJECTS (RO)

Concept in CM	Concept in RO	Mapping Type
core:Process	wfdesc:Process	Equivalence
core:Aggregation of Processes	wfdesc:WorkflowTemplate	Equivalence
core:Business Object	wfdesc:Input, wfdesc:Output	Equivalence (inferred)
core:Business Object	wfdesc:Parameter	Super-class
premis:Format	wfdesc:Artifact	Equivalence
janus:workflow_run	wfprov:WorkflowRun	Equivalence
janus:processor_exec	wfprov:ProcessRun	Equivalence
core:Artifact	wfprov:Artifact	Equivalence
core:SystemSoftware	wfprov:WorkflowEngine	Super-class
core:Process & core:Service & core:Interface	wf4ever:WebServiceProcess	Equivalence (inferred)
core:Artifact (& premis:Format)	wf4ever:Dataset / wf4ever:Document / wf4ever:File / wf4ever:Image	Superclass / Equivalence (inferred)

3) Execute regression: This process executes the regression parametrised in the regression control, using the data-set generated in the extract data process
4) Generate aggregation: since a dam has a large number of sensors and a regression is used for each physical quantity associated with each sensor, we need to run hundreds or thousands of regressions. Thus, the process aggregates all MLR executions into one report. This step generates the controls that define how this data is aggregated
5) Produce report: collects all the results and compiles them into a single report

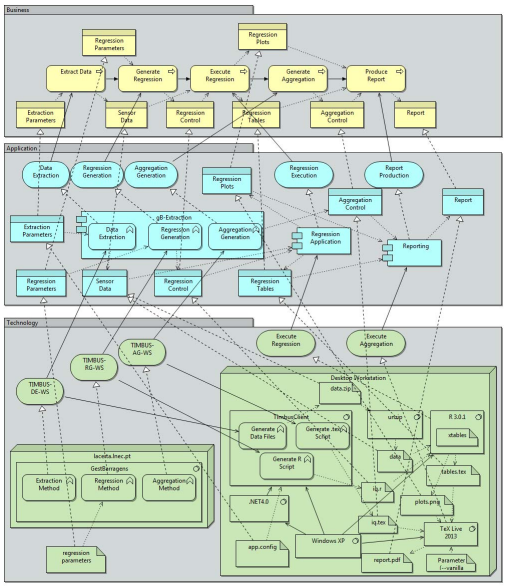

Fig. 6. Structural safety data analysis process

The "Technology" section of Figure 6 depicts the technological infrastructure that supports the execution of this process, using only concepts from the core ontology of the Context Model. We can identify two different computing platforms, a server running the "GestBarragens" application that collects and stores the data, and a Desktop workstation that is used for the data analysis. A web service provides the data, and the "TimbusClient" is utilised to connect to it, and generate scripts based on the downloaded data. The client needs the .NET 4.0 framework, which itself requires Windows; in our specific example this is satisfied by using Windows XP. The scripts for model building are executed in the "R" environment, specifically version 3.0.1 is utilised. Finally, the report is generated using Latex, provided by "TeX Live 2013", and the a specific style file.

The second use case is a scientific experiment in the domains of machine learning and music information retrieval. The experiment performs an automatic classification of music into a set of predefined categories, and evaluates the performance of this classification against a ground-truth (gold standard). The experiment involves several steps:

1) Music data is acquired from online content providers.
2) For this music data, genre assignments (the ground-truth) are obtained from websites such as Musicbrainz.org.
3) A web-service is employed to extract numerical features describing certain characteristics of the audio files.
4) The numerical description and the genre assignments are combined, and a machine learning model is trained and employed to predict genre labels for unknown music.
5) Finally, the performance of this prediction is assessed.

Besides these steps, several scripts are used to convert data formats and for other similar tasks. The use case is described in more detail in [23]. The specific instance of the use case was originally executed in a Linux environment issuing commands via the system shell, with the researcher ensuring the correct orchestration of steps in the correct order. For various analysis purposes, this workflow has been converted to an implementation using the Taverna workflow engine, and has also been converted into a research object [24]. A model of the process using the core concepts of the Context Model is depicted in Figure 7. Regarding technology, we can observe the Taverna workflow engine, which depends on a Java Virtual machine 6.0, which in turn has been executed on version 11.04 of Ubuntu Linux on a 64-bit workstation. We further see license and patent information for the software components. Figure 8 then depicts a section of the technology layer, augmented by elements from the PREMIS, Software License and Patents ontology. We can identify details on the data formats of the artifacts, such as MP3 or ARFF.

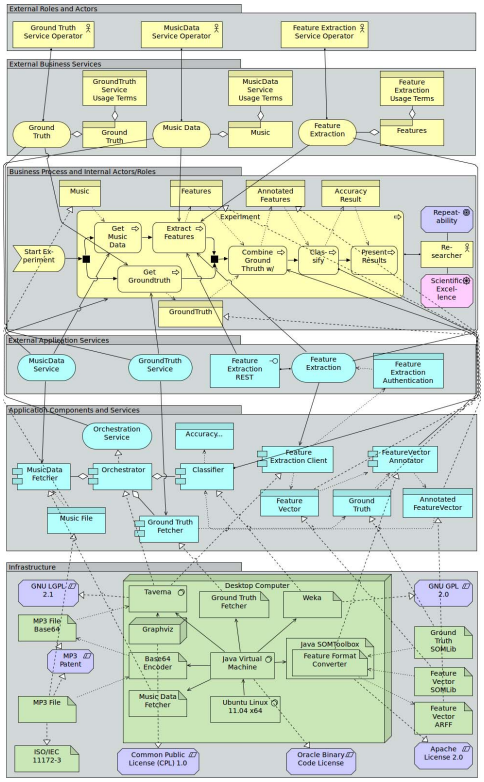

Fig. 7. Music classification process – core concepts

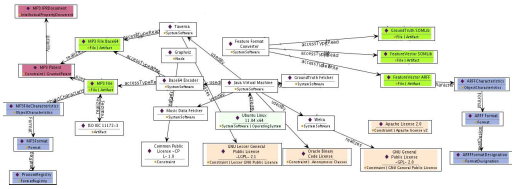

Fig. 8. Music classification process – usage of extension ontologies

VI. Conclusion

In this paper, we presented the TIMBUS Context Model for process preservation, the architectural principles and the core and extension ontologies. We presented a detailed analysis of the similarities and differences, and expressive power of Research Objects and the Context Model, and demonstrated how one can be transformed into the other. This is the basis for a more tighter integration of the two approaches, to augment and refine their respective concepts. We finally showed how the Context Model can be applied to two use cases in the domain of data analysis. Future work will focus on providing examples how existing instances of each model can be refined and augmented in their description and expressiveness by utilising the mapping between the model and detailing the concepts in the other ontologies. A range of tools is under development to automate parts of the creation of a Context Model, e.g. by detecting installed software and their licenses, hardware specifications, or communication to external services.

References

[1] J. Zhao, J. M. Gómez-Pérez, K. Belhajjame, G. Klyne, E. García-Cuesta, A. Garrido, K. M. Hettne, M. Roos, D. D. Roure, and C. A. Goble, "Why workflows break - understanding and combating decay in taverna workflows," in *8th IEEE International Conference on E-Science (e-Science 2012), Chicago, IL, USA, October 8-12, 2012*. IEEE Computer Society, 2012, pp. 1–9.

[2] C. Webb, *Guidelines for the Preservation of Digital Heritage*, S. Information Society Division United Nations Educational and C. O. (UNESCO), Eds. National Library of Australia, 2005.

[3] D. B. Marcum, "The preservation of digital information," *The Journal of Academic Librarianship*, vol. 22, no. 6, pp. 451 – 454, 1996.

[4] S. Granger, "Emulation as a digital preservation strategy," *D-Lib Magazine*, vol. Vol. 6 (10), 2000.

[5] S. Higgins, "The DCC curation lifecycle model," *The International Journal of Digital Curation*, vol. 3, pp. 134–140, 2008.

[6] H. Brocks, A. Kranstedt, G. Jaschke, and M. Hemmje, *Smart Information and Knowledge Management*. Springer, 2010, ch. Modeling Context for Digital Preservation, pp. 197–226.

[7] E. Conway, B. Matthews, D. Giaretta, S. Lambert, and M. Wilson, "Managing risks in the preservation of research data with preservation network," *International Journal of Digital Curation*, vol. 7, 2012.

[8] K. Belhajjame, O. Corcho, D. Garijo, et. al, "Workflow-centric research objects: First class citizens in scholarly discourse," in *Proceedings of Workshop on the Semantic Publishing, (SePublica 2012) 9th Extended Semantic Web Conference*, May 28 2012.

[9] H. Van de Sompel and C. Lagoze, "Interoperability for the Discovery, Use, and Re-Use of Units of Scholarly Communication," *CTWatch Quarterly*, vol. 3, no. 3, August 2007.

[10] P. Ciccarese, M. Ocana, L. Garcia Castro, S. Das, and T. Clark, "An open annotation ontology for science on web 3.0," *Journal of Biomedical Semantics*, vol. 2, no. Suppl 2, p. S4, 2011.

[11] D. De Roure, K. Belhajjame, P. Missier, J. Manuel, R. Palma, J. E. Ruiz, K. Hettne, M. Roos, G. Klyne, and C. Goble, "Towards the preservation of scientific workflows," in *Procs. of the 8th International Conference on Preservation of Digital Objects*, Singapore, 2011.

[12] M. Lankhorst, *Enterprise architecture at work: modelling, communication, and analysis*. Springer, 2005.

[13] W3C, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)," Tech. Rep., December 2012.

[14] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.

[15] M. Rosemann, P. Green, and M. Indulska, "A reference methodology for conducting ontological analyses," in *Conceptual Modeling–ER 2004*, ser. Springer Lecture Notes in Computer Science, 2004, vol. 3288.

[16] T. O. Group, *ArchiMate 2.0 Specification*. Van Haren Publishing, 2012.

[17] R. Treinen and S. Zacchiroli, "Description of the CUDF Format," Tech. Rep., 2008, http://arxiv.org/abs/0811.3621.

[18] PREMIS Editorial Committee, "Premis data dictionary for preservation metadata," Tech. Rep., March 2008.

[19] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere, "Towards an extensible context ontology for ambient intelligence," in *Ambient intelligence*. Springer, 2004, pp. 148–159.

[20] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, "Taverna, reloaded," in *Proceedings of the 22nd international conference on Scientific and Statistical Database Management*. Springer, June 2010.

[21] S. S. Sahoo and A. Sheth, "Provenir ontology: Towards a framework for escience provenance management," Wright State University, Tech. Rep., 2009.

[22] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, and A. Caetano, "Using Ontologies for Enterprise Architecture Integration and Analysis," *Complex Systems Informatics and Modeling Quarterly*, no. 1, 2014.

[23] R. Mayer and A. Rauber, "Towards Time-resilient MIR Processes," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, October 8-12 2012, pp. 337–342.

[24] K. R. Page, R. Palma, P. Hołubowicz, G. Klyne, S. Soiland-Reyes, D. Garijo, K. Belhajjame, and R. Mayer, "Research objects for audio processing:capturing semantics for reproducibility," in *Proceedings 53rd AES Int. Conference on Semantic Audio*, London, January 2014.