Federating real world information about business processes into the TIMBUS Context Model

Dr. Till Riedel, Karlsruhe Institute of Technology

The TIMBUS Context Model is a core part of the architecture that we have used to model and analyse real world business processes. This white paper gives a high level overview on how linked data structures can be used to federate relevant manually modelled and automatically extracted information into a single consolidated model for analysis and digital preservation. Most of the software described in this white paper is available as open source and allows preservation tool chain developers to consider dependencies on third-party services, information and capabilities that will be necessary to validate digital information in any future usage context.

1 Introduction

As many artefacts and activities are moved into a digital world, digital preservation undergoes new challenges and opportunities. Rather than to stop with preserving a single artefact together with rather static meta-data it becomes more and more relevant to preserve the processes that allow the interpretation of and operation with artefacts and data. For this not only the abstract (domain independent) definition of those processes is important to capture but the concrete context of a business process or service. The TIMBUS Context Model is a modelling tool and data structure that allows capturing both domain specific and independent models of real life business processes or services.

The terminology “context of a business process or service” refers to the physical, digital and social environment of an instance of the abstract concept of a business process or service. Similarly, the terminology “context model of a business process or service” refers to any modelling that can be used to document, abstract and characterize the physical and social situation of an instance of the abstract concept of a business process or service. The terms “Model” and “Ontology” are used in an interchangeable manner within this scope. The model or ontology generally consists of a conceptual level (a “meta-model”) and an instance level (the concrete context captured for a use case).

The TIMBUS Context Model assumes central importance in the preservation of business processes, providing a means of modelling context and dependencies so that all the information required for preserving and redeploying a process is captured. Within the project we particularly have adopted a common core Domain Independent Ontology (DIO) based on the Archimate standard (http://pubs.opengroup.org/architecture/) that provides many opportunities by providing a low entrance barrier to DP from the architecture community, and allows getting agreement to concepts on an abstract level.

The use cases explored within the project go beyond modelling architectures towards analysing contextual risks and link resources for concrete preservation and redeployment actions, along with risk and preservation-relevant-context, internal and external to both the use case and the controllable domain of an entity. Furthermore, it is often unknown which aspects and detail will be relevant for future preservation actions. The result of 3 years work is a comprehensive model developed based on best-practices, standards, and industrial case stakeholders’ requirements, with an extensible architecture and a governance method for evolving the model and adapting it to diverse preservation scenarios. The current extensible TIMBUS Context Model, as pictured in **Error! Reference source not found.**, supports reasoning and inference, which can be used for checking inconsistencies on the model and inferring information that might be particularly useful for the TIMBUS preservation processes.

As figure 1 shows, in contrast to other approaches, the TIMBUS Context Model is not monolithic but follows a linked data approach that allows to include various sources of information and various view-points to comprise different aspect of real-life business systems and challenges on performing preservation actions on them.

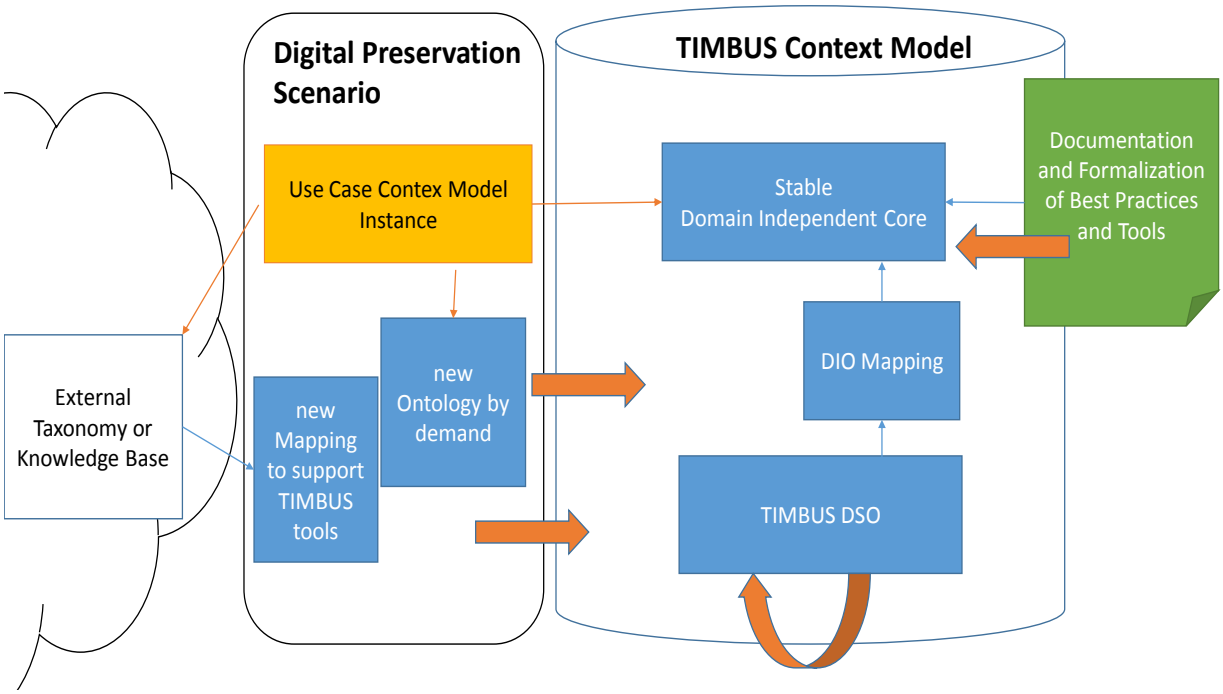


Figure 1: Building a Context Model for your Use Case

2 Reasoning on the Context Model

One of the main capabilities of the context model is that it allows (semi-)automatic analysis of the data captured. Those capabilities can be subsumed under the term “reasoning” and allow the user and other system components to query connections between entities, discover new facts and inconsistencies in the modelled data based on declarative rules. While the context model supports different kinds of reasoning, the use of description logic for computational inference based on the second version ontology web language (OWL 2) is at its core.

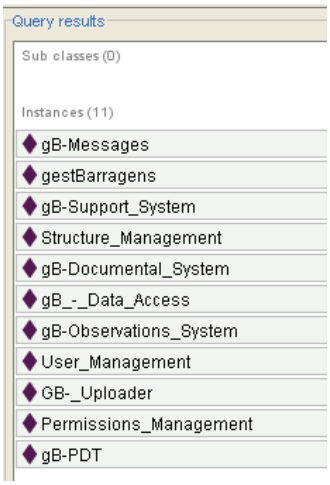
By using the reasoning capabilities, the correctness of the ontology can be validated and questions such as the following can be answered:

1. What *BusinessServices* are used by the *BusinessRole* <*Customer*>?
2. What are the Debian *Packages* the *BusinessProcesses* X dependsOn?
3. What *SystemSoftware* is realized by a certain File *Artifact* on this Linux *Node*?

TIMBUS takes from the computational inference features of ontologies for performing business process analysis. By using a Description Logic reasoner, features such as consistency checking, dependency inferring, and completeness checking are available, thus allowing the checking of the correctness of the EA models expressed on the DIO and DSOs. Querying facilities are also available for some ontology languages, which makes possible the retrieval of data stored in the ontologies and even simple computations by the application of filters.

The different reasoning configurations can be used for the purpose of identifying the dependencies between elements more easily. For instance, the question "*what are the technological entities supporting the <temperature monitoring>?*" can be translated into the description logic query:

hasLayer some TechnologyLayer and dependsDown value <temperature monitoring>



Such a query results in a list of entities that might be relevant to preservation activities around <temperature monitoring>. TIMBUS for sure will not try to preserve the monitored structure, that a sensor in this process was attached to, but it will give access to a large set of federated information that is relevant to the preservation process. It further provides the means to analyse this data set under specific aspects. The execution of these queries (that can be tried out using the Protégé Desktop on the example processes on http://timbus.teco.edu/ontologies/examples) will thus highlight the dependencies between the different elements of the infrastructure, and can be used as a valuable tool for decision making. By identifying the dependencies, it is possible to trace the propagation of the changes throughout the architecture. That information can then be used by the organization for decision making. Moreover, since the architecture can be enriched with the addition of new context information sources, meaning other types of models, it becomes possible that other kind of decision-making analysis can be performed. The TIMBUS Context model is thus the basis for the holistic preservation approach.

3 Mapping heterogeneous preservation aspects to a common Context Model

The TIMBUS context model is not building on a single model approach, but is able incorporate notations and existing modelled knowledge into a common model for performing informed preservation actions.

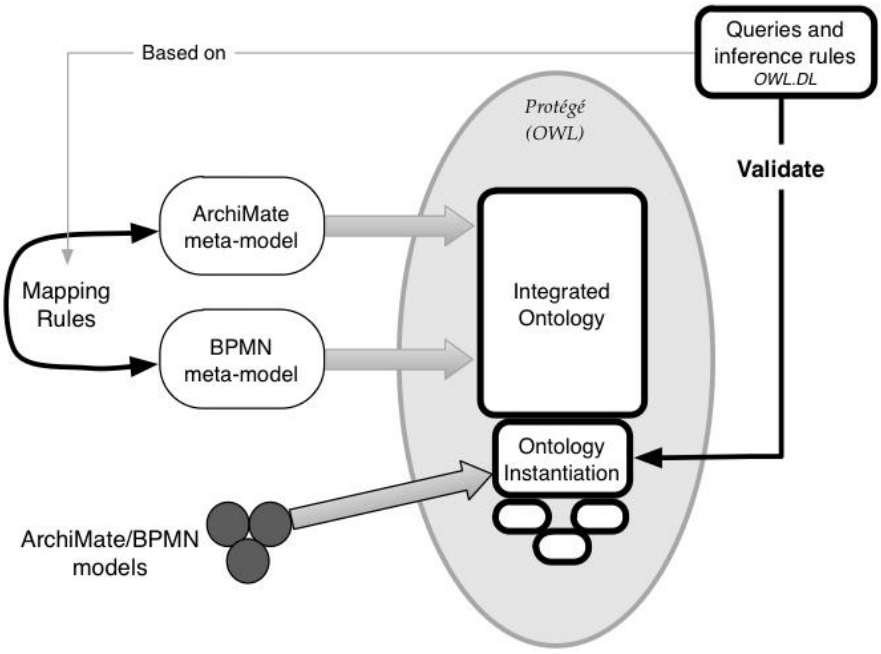


Figure 2: Ontology mapping scheme for BPMN Models

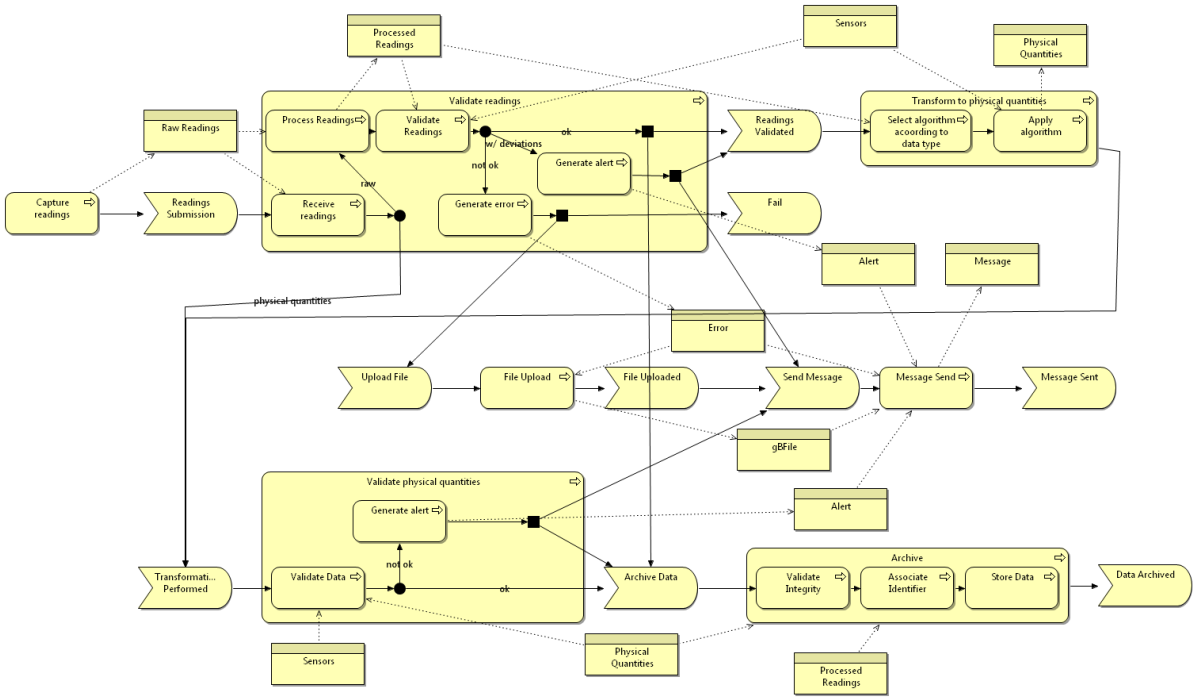


Figure 3: Business Process manually modelled with Archi (http://www.archimatetool.com/)

The core of the TIMBUS context model is one upper ontology that was modelled on the basis of the Archimate 2.1 specification (http://pubs.opengroup.org/architecture/archimate2-doc/) by the Open Group. It contains core elements as typically used in any real world system based on a generic architectural perspective. This model, which serves as domain independent ontology, links the different domain specific ontologies together based on:

- Axiomatic interlinking (ontology concept mapping) using description logic reasoning
- Source level linking through persistent unique identifiers (references)
- Import time linking using format conversion (transformation)
- Identity discovery on the heterogeneous linked data (instance mapping)

One example of an existing standard that we successfully mapped on ontology level is the Business Process Model and Notation (BPMN), which is used in many industries to represent activities, control flows, data, and auxiliary information about a process as a so called a Business Process Diagram (BPD).

TIMBUS represents BPMN meta-models into ontologies (OWL), and then makes an integrated ontology by mapping the common concepts and relationships of them by the use of ontology mapping techniques. Mappings have been developed within the project to overcome syntactic, lexical and semantic mismatches between the models.

The TIMBUS context model is also building strongly on best practices and established models in the preservation and business community that are mapped into the context model. In a business process, a number of digital objects are created, modified or read. Information on the format of these objects is crucial for any preservation action to be carried out, as e.g., migration of data files to a different format might require changes in the rest of the process. Besides having more impact on subsequent processing steps, in regards of formats, the scenario of a business process is not much more complex than in traditional digital preservation settings.

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 established PREMIS Data Dictionary (http://www.loc.gov/standards/premis/), which is also available in the form of an ontology and mapped that into the Archimate-based TIMBUS DIO.

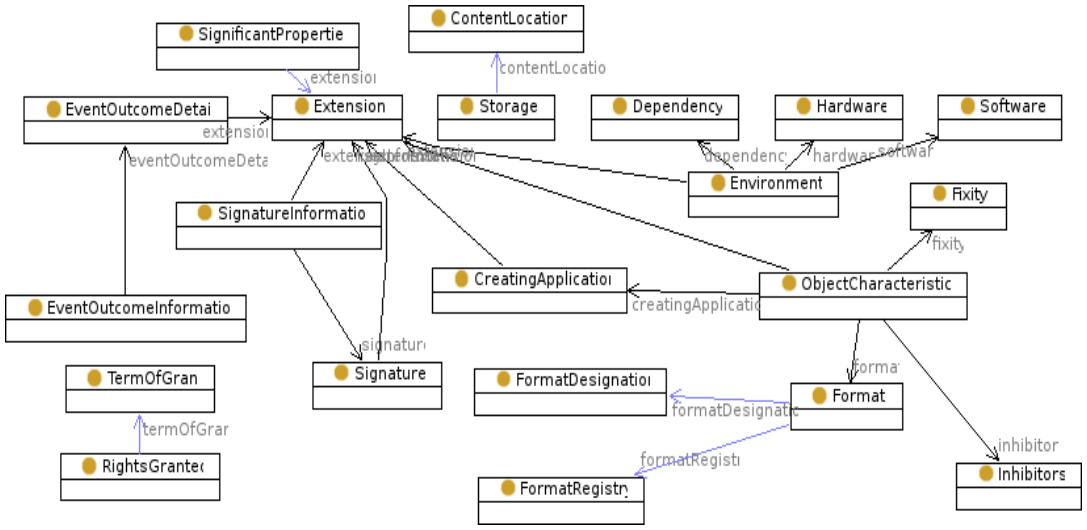


Figure 4: Relevant parts mapped parts of the PREMIS ontology

4 A model that fits its users

While technical requirements where one driver, the TIMBUS context model was developed with the users in mind. The project included experts from 3 different application domains and explored real life use cases (see ontology examples on http://timbus.teco.edu/ontologies/examples). During the development of the context model we incorporated user feedback into both the construction of the model and the development of the tools.

We are confident that the TIMBUS modelling flow can be adopted in many application areas. Especially, it was suggested that the process could be done quick after an initial learning curve. One of the experts we interviewed stated, while refinements on the initial model were quite time consuming: “The [new] model was then done in 1-2 hours, because I knew what I should exactly model and what [concepts] were present in [the DIO].” Although the TIMBUS context model is the output of a research project and still continuously matures, most experts were generally satisfied: “These concepts clearly have their justification, if we clearly specify them: This all makes sense!”

5 Automatically extracting the preservation context from real systems

We have argued that there are many aspects in the context of a business process that have to be taken into account during preservation planning and execution, to ensure successful re-deployment of that process. In the context of a business process, (1) there are abstract (coarse-granular) aspects which are relevant to the entire domain of process preservation, and (2) there are more specific aspects (fine-granular) which are relevant to sub-domains of process preservation, e.g. the class of scientific processes or an individual scientific experiment, which may identify further relevant aspects.

Particularly the fine-granular knowledge is subject to frequent changes during the life-time of a process and is difficult to capture manually. Being able to capture the complete knowledge, however, on dependencies accurately between components is important to enable the successful re-deployment of a preserved business process in future. With respect to computational environments, the dependencies between software and hardware components need to be accurately capturable in their entirety, such that an informed decision about what to preserve can be made.

In the extraction phase, the target business machines' relevant information is captured by means of using numerous different extractors, each of which will have its own processing depending on the relevant information and discipline they want to capture. For example, a business may have the need to capture information from Linux packages, hardware, Perl modules, network, and legalities, while others may run on a Windows environment, using dedicated platforms and legacy applications. The added value of having an open framework is to allow multiple implementations of extractors to be aggregated and consolidated in doing the best capturing of the target system according to the business needs. On the other hand, a business running on top of a pool of VMs may not have any relevant need regarding the hardware. The set of extractors is tailored to the specific needs of the preservation scenario.

5.1 Process extractor and monitor

The process extractor mines business process models from event logs, using an optimisation method and conformance checking techniques for finding business process models which best comply with a given event log. The models and optimisation techniques are described in more detail in TIMBUS Deliverable D6.5. Various extensions have been developed to enable use case specific mining, as well as filtering and noise reduction. The latest version enables extraction from logs and population of the TIMBUS Domain Independent Ontology (DIO).

Furthermore, the latest version contains a conformance checking plugin which allows for the evaluation of process models with regards to a given event log. This plugin can be used to import and evaluate external processes, as well as the models mined by the optimisation techniques. Whereas the previous version relied on extensive simulation and log generation for determining model conformance, the new plugin works directly on the process model. It uses algorithms which extract activity relations from the model, and compares these with the activity relations present in the given event log. In addition, an algorithm for fast replay of event logs on business process models has been integrated into the tool.

As opposed to the Process Extractor, the Process Monitor is working in a continuous fashion and is not statically executed or invoked. Generally speaking, it monitors an event stream and processes the events into an "online" process footprint in a dynamic fashion. In a second step this footprint can then be interpreted into a business process model conforming to BPMN terminology. Additionally to the evolutionary approach that is used by the static Process Extractor - which is preferably for an initial business process discovery - a deterministic approach called Constructs Competition Miner (CCM) has been developed. The CCM is a process discovery algorithm that follows a divide-and-conquer approach to directly mine a block-structured process model which consists of common BP-domain constructs and represents the main behaviour of the process.

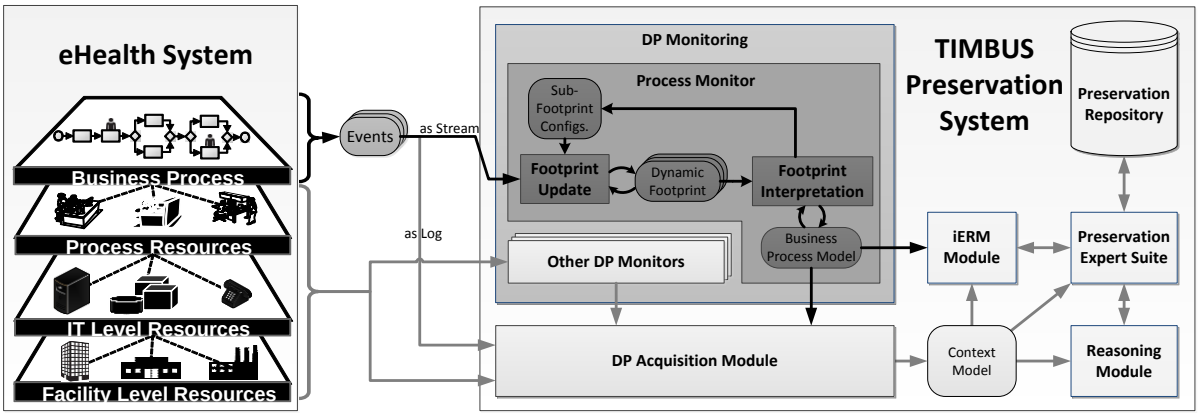


Figure 5: Application of the process monitor in the SAP eHealth Scenario

5.2 Software Dependency Extractors

The Debian Software Extractor is a tool that allows extracting information from software packages and corresponding interrelations in a Debian-based Linux distribution. Such distributions are built on top of a low-level package manager – called dpkg – which allows installing, removing and upgrading local software packages, and also feature a higher-level package manager – called apt – which allows doing the transitive installation and configuration of software packages and dependencies, as well as allowing installation from remote locations. This extractor uses both dpkg and apt, in order to gather all meta-data, both from the installed packages and from the packages known to the system's package managers, into a JSON format. It not only allows extracting package information both locally and remotely but also, in addition to the meta-data, extracts the licenses and the packages' installer remote locations. The resulting extraction is afterwards sent for conversion into an ontology instance, keeping the original interrelations and relevant information. The resulting conversion is then preserved for later querying or visualization.

Another dependency extractor is the Perl Modules Extractor that allows to analyse and report which Perl Distributions and dependencies exist in a specific machine. Although this could be accomplished through several methods, we chose an approach that utilizes Perl mechanisms to infer Perl Modules requirements. Using the keys from the dependency modules hash table, each package is checked individually to decide to which Distribution they belong to. This check is performed through the MetaCPAN (https://metacpan.org/) federated Perl package knowledge base. Just like in the Debian package case the output conforms to the Common Upgradeability Description Format (CUDF) ontology and maps directly into the TIMBUS Context Model, allowing the user to treat both types of packages in the same way.

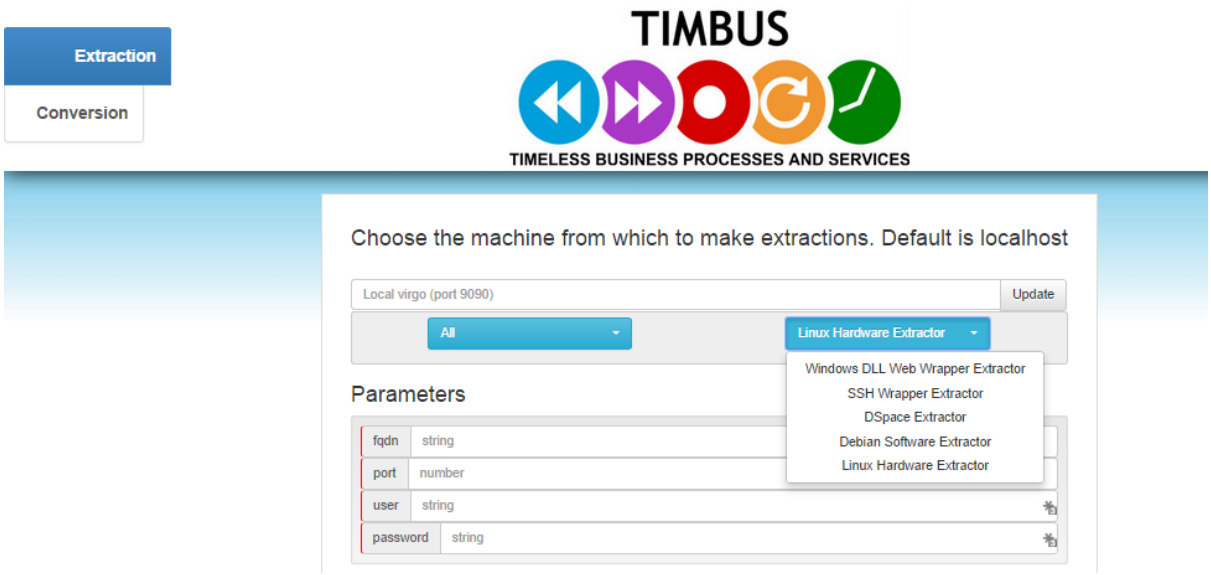


Figure 6: Extraction GUI as presented towards the digital preservation expert

The extraction process can be configured by a web application that also filters the existing dependency extractors according to other criteria, e.g., by OS. Supposing that one of the machines to be extracted is Windows-based, it may be interesting to list only the extractors that are applicable to this OS, as can be seen on Figure 5. This is all dynamic information that is provided by each extractor.

5.3 Network Dependencies Monitor

A local-remote (lr) network dependency refers to the situation between a client that requires a server to provide its own service to another client. For example, a web server may require a database to service web browsers a web service. Therefore, the web server lr-depends on the database. Aremote-remote (rr) dependency refers to the situation when a server requires a client to call another server before calling itself. For example, a web server may require a web browser to call a name server first. Therefore, the web server rr-depends on the name server.

The network dependencies monitor is a system for capturing direct lr-dependencies between programs and hosts, and inferring indirect lr-dependencies from the direct ones. Indirect dependencies are given by the transitive closure of direct ones. The system captures every network interaction between processes to directly detect lr-dependencies between these processes. Furthermore, if two processes A and B, and two processes B and C are detected to be dependent at the same time, the two processes B and C are assumed to be dependent too. Dependencies between processes determine dependencies between their programs and between their hosts. Given that the capturing process has seen the complete set of program traces, this approach captures any lr-dependencies between programs and hosts.

Our system features various possibilities to monitor lr-dependencies in real Linux environments:

- **SystemTap**[1]: We hook into the networking stack of the Linux kernel using SystemTap to track any connections on the level of TCP.

[1] http://sourceware.org/systemtap/

- **NetStat**: Polls kernel published connection tables in order to get per application network connections
- **Apache Logging**: Instruments and redirects the Apache webserver logging configuration to mine connections to web service on the server

The stream of events from logging events is continuously transformed into instances of the TIMBUS DIO ontology[2]. Figure 7 shows the relevant part: any network connections between local and remote networking interfaces are represented in the ontology and related to each other. Furthermore, the host of local network interfaces is represented and the local programs that have initiated connections are documented.

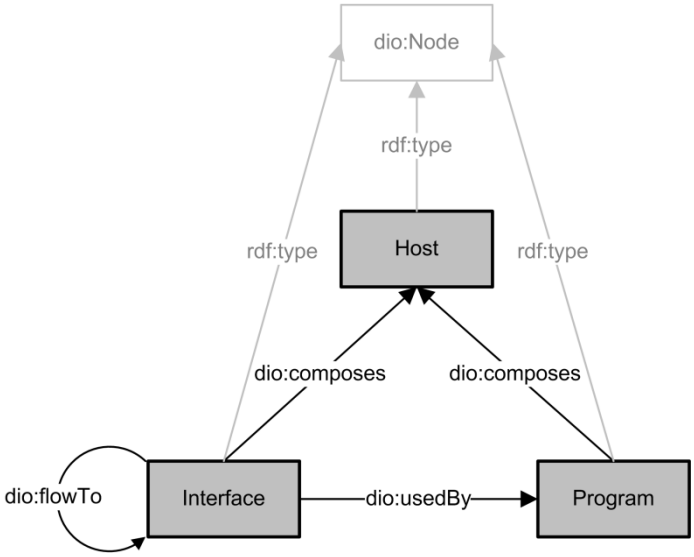


Figure 7: Network Dependencies Ontology

Modern business processes form considerably complex dynamic ecosystems. A business process may span many involved legal parties, interacts with many people having varying roles, concerns, responsibilities and authorizations, and is supported by a complex distributed computational environment.

5.4 Large-scale Integration of extracted context

TIMBUS extractors like the ones outlined above easily generate a large knowledge base of facts, e.g. about 50.000 facts for a relatively moderate single machine use case. Particularly the integration of the manually modelled ontologies and the extracted parts of the context model seems unmanageable for manual integration. As the problem is common in heterogeneous linked data applications, we did not want to build a custom solution that deemed us as not sustainable given the interest of the partners. On the other hand the integration with the TIMBUS architecture and the preservation workflow was important.

The Linked Data Integration Framework (LDIF) framework that is actively developed by a group around Christian Bizer at University of Mannheim integrates well with all TIMBUS tools, which was a large part of the considerations when selecting the framework for inclusion. One important aspect is that it will

[2] https://timbus.teco.edu/public/ontologies/dio.owl

also add scaling out support to the integration process. The new Hadoop-based implementation of LDIF provides for processing very large data sets on a Hadoop cluster, either on a common cloud provider or a private cloud. This allows natural scaling with larger IT-landscapes for the context population phase. TIMBUS thus decided to adapt this integration strategy following a classical Extraction-Transformation-Loading (ETL) scheme.

- **Extract**: The Import modules retrieve and replicate data sets that are collected by the extractors via file download, crawling or SPARQL. For TIMBUS we extended this framework by multiple possibilities of file conversion on data ingress. It directly interfaces with the different TIMBUS extractors. LDIF already supports a number of data formats used by TIMBUS and externally developed knowledge extractors. The triple/quad dump importer allows importing of RDF/XML, N-Triples, N-Quads and Turtle. This suits the importing of all TIMBUS extractors, which natively export ontology facts in the above formats. Furthermore SPARQL importing is supported for remote access to existing endpoints.
- **Transformation**: The LDIF SILK identity resolution component discovers URI aliases in the input data and replaces them with a single target URI based on user-provided matching heuristics. This part is of particular interest in TIMUS as manually modelled entities are only mapable to extracted entities by humans. With SILK this task can be automated over multiple extractions.
- **Loading**: LDIF outputs the integrated data are directly written to a Fuseki graph store which then also runs a reasoner based on the integrated and cleaned up ontology. Beyond purely merging different extractions on instance level the framework has also built-in capabilities for keeping track of data provenance. It retains the original extraction context as a named graph.

We provide a TIMBUS specific GUI to LDIF that allows us to integrate extracted TIMBUS context information into an instance of the TIMBUS DIO. The tool is a user interface to the Linked Data Integration Framework (LDIF) with TIMBUS-specific data import and integration building blocks. The tool is delivered as a web application that has been developed on top of SAP UI5 (now OpenUI5).

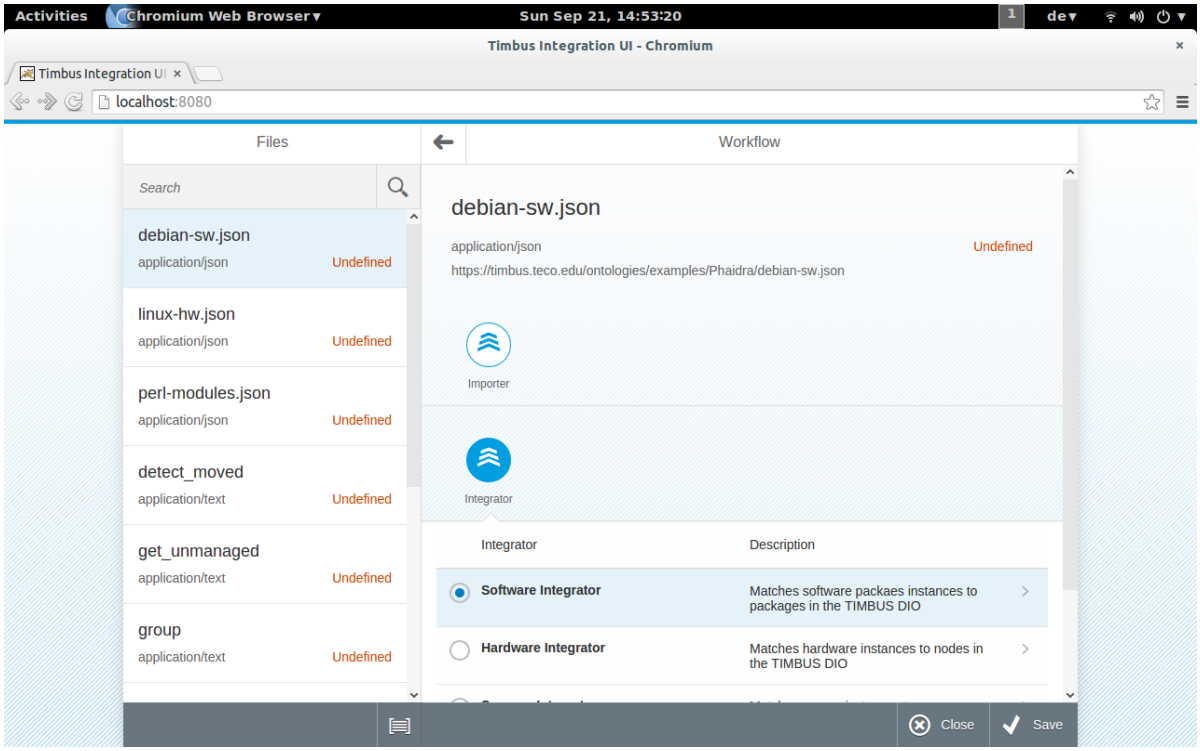


Figure 8: Integration Job that semantically merges extracted software with the Archimate model

Natively LDIF supports file based ontologies, web crawlers and SPARQL Endpoints to existing knowledge basis. Because many extraction tasks involve the “semantic lifting” from simple structured documents like UNIX command output or configuration files, we extended one of the new importers to allow loading of both structured text and XML documents and transforming it consecutively via a XSLT tool chain. The importer further integrates with an XSLT engine into LDIF so that a user can easily extend the context model using simple declarative style sheets. TIMBUS provides multiple predefined style sheets for various kinds of conversions tasks. Examples are the extraction of *UserRoles* from UNIX “/etc/passwd” and “/etc/group”, or the parsing of local modifications within subversion file repository structures. h

The transformation step purely works on top of the existing LDIF capabilities exposing SILK (http://silk.wbsg.de) and R2R (http://r2r.wbsg.de/) via a simple end user targeted interface that allows the user to select specific predefined integration flows. One example is the integration of automatically extracted package information with the models converted by Archimate. A precondition for this mapping is the semantic mapping of the concepts used the different models (Archimate and CUDF). Within the sets defined by semantically related types (equalities and subsets) we use SILK to identify equal entities by stochastic matching (here based on package names) to automatically interlink the models on instance level. The GUI outputs ETL schedules that can be automatically executed by a server to provide always up to date context information.

Beyond the server based automatic tool chain, all tools are available to be used locally on a single machine for easy further development and processing using standard ontology tools like Protégé (http://protege.stanford.edu/). Newly developed visualization components particularly make it easier to handle the converted and extracted information. The project tool box also combines existing tools e.g. for extracting the so called syntactic locality module (only information connected to a certain process) and to convert the output into web-friendly formats like JSON-LD.

6 Conclusion: Bringing the TIMBUS Context Model into context

The TIMBUS project has designed and evolved a Context Model with versatile applicability to many real world use cases. It allows modelling the relevant context of a business process that is to be digitally preserved. The population and modelling framework as shown in Figure 9 already features a large set of plugins to populate the content of a model instance, that were necessary to satisfy the diverse needs of the TIMBUS use cases. In addition, the framework is extensible for new data sources and plugins. While we do not expect that all domain specific models or extraction tools will be usable in other use cases, the large set of open source tools will allow others to build upon the experiences and apply the TIMBUS methodology. The Context model as also shown in the figure integrates many population and analysis tools at the core of the TIMBUS Architecture. It allows interoperability and data exchange in a well defined manner fulfilling different preservation needs.

Beyond pure data exchange the model is flexible and "smart" enough to be future proof and evolvable: The reasoning capabilities inside the model and the framework lets tools adapt to data from different the federated information integrated from various heterogeneous data sources. Future developers are invited to build upon the tools published and maintained by TIMBUS partners on http://opensourceprojects.eu/p/timbus and to look at the large set of end user and training material provided on http://www.timbusproject.net.

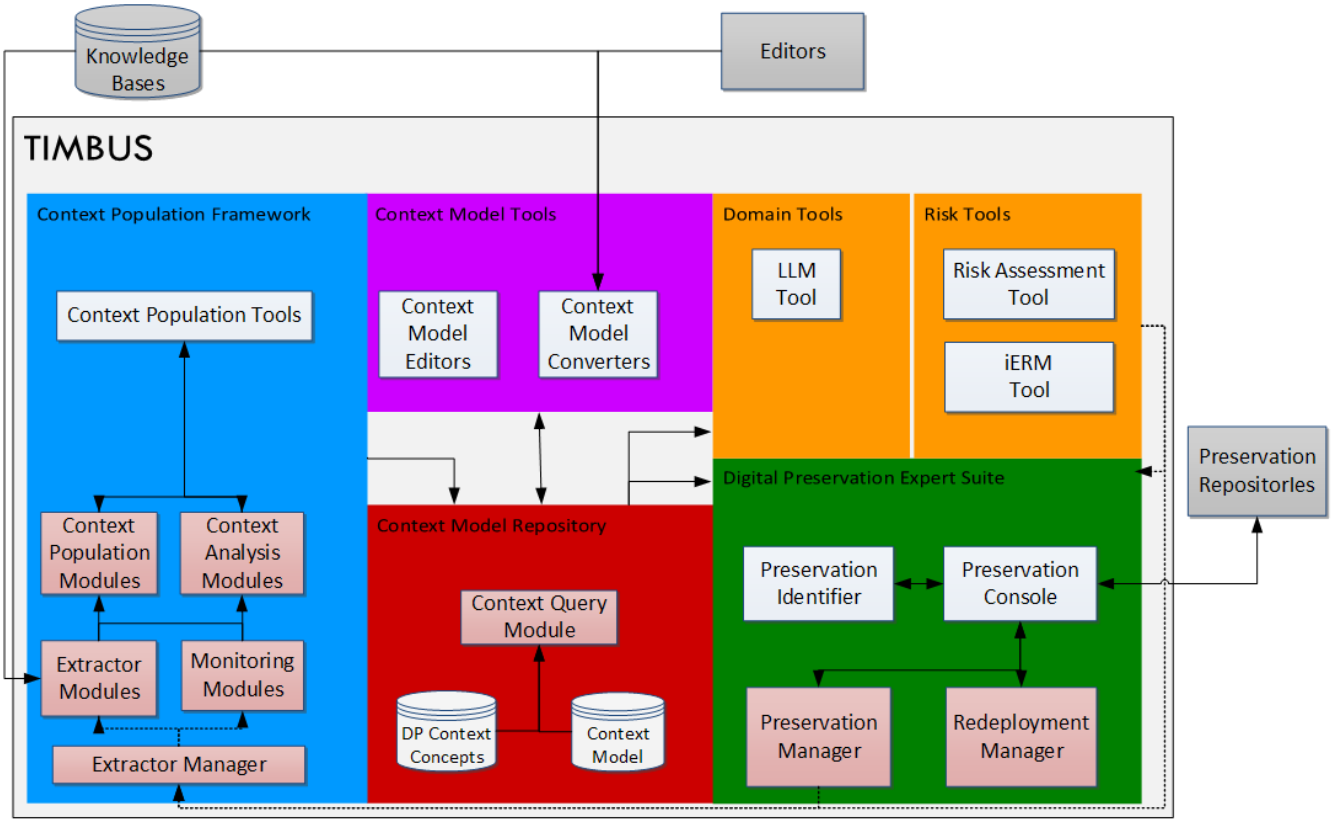


Figure 9: Simplified Overview of Tools on http://opensourceprojects.eu/p/timbus

7 References and Further Reading

Figures and text has been partially taken from the following publications, that also provide a good starting point to further insights concerning the scientific background of the work:

- TIMBUS Public Project Deliverables D4.3_M24, D4.9, D6.10 (accessible via http://timbusproject.net)
- Antunes, G., Caetano, A., Bakhshandeh, M., Mayer, R., & Borbinha, J. (2013, January). Using Ontologies to Integrate Multiple Enterprise Architecture Domains. In *Business Information Systems Workshops* (pp. 61-72). Springer Berlin Heidelberg.
- Jardim-Goncalves, R., Coutinho, C., Cretan, A., da Silva, C. F., & Ghodous, P. (2014). Collaborative negotiation for ontology-driven enterprise businesses. *Computers in Industry*.
- Molka, T., Redlich, D., Drobek, M., Caetano, A., Zeng, X. J., & Gilani, W. (2014, March). Conformance checking for BPMN-based process models. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (pp. 1406-1413). ACM.
- Neumann, M. A., Riedel, T., Taylor, P., Schmidtke, H. R., & Beigl, M. (2011). Monitoring for digital preservation of processes. In *Modeling and Using Context* (pp. 214-220). Springer Berlin Heidelberg.
- Neumann, A., Miri, H., Thomson, J., Antunes, G., Mayer, R., & Beigl, M. (2012, October). Towards a decision support architecture for digital preservation of business processes. In *Proceedings of the 9th Int. Conf. on Digital Preservation (iPres 2012)*.