Knowledge-Intensive Process Modelling in Engineering Design[†]

Wolfgang Mayer Arndt Mühlenfeld Markus Stumptner
University of South Australia
Advanced Computing Research Centre
mayer|arndt.muehlenfeld|mst@cs.unisa.edu.au

Abstract

While design processes in certain domains have shifted towards early adoption of simulation and virtualisation techniques, processes monitoring and reuse is not well-integrated into current development practices. We introduce a framework to integrate Multidisciplinary Design Optimisation processes using ontological engineering, where artefact and simulation models are exploited to yield more effective optimisation-driven development. We show how meta-modelling techniques can overcome representational and semantic differences between analysis disciplines and execution environments.

1. Introduction

In the design and engineering context, ontologies provide an explicit formalisation of design knowledge that is otherwise distributed among several teams [9]. Ontologies also aid in semantic interoperability between design disciplines due to the introduction of meta models that serve as a linking element between disciplines [14], providing means to reason about process-, simulation- and domain-specific aspects [3].

As designs become more complex, designers and engineers increasingly rely on tool support to manage not only design artefacts, but also the design processes themselves. In order to store, manipulate, connect and validate processes, semantic representations of processes are desired that are able to convey not only the structure, but also the semantics of process parameters and activities unambiguously. This has become an even more pressing issue since many global manufacturers increasingly rely on distributed supply chains where consumer-specified manufacturing and machining processes may be imposed on suppliers.

While research in distributed scientific computing [18, 13] and Web Services has led to numerous attempts to represent the "meaning" of individual tasks, most frameworks do not fully address the challenges posed by non-trivial design tasks, where information is rich in structure and detailed interpretation of semantics is necessary. Conversely, static artefact models fail to capture dynamic aspects of processes and their execution. Hence, effective tool support requires an *integrated* approach that facilitates the development of adequate models of *individual artefacts and tasks*, *processes* and their *execution* [15].

The work described in this paper extends the emphasis from artefact-centric approaches to *process-oriented* ontologies to provide a comprehensive unified framework. We outline an ontological representation of typical multidisciplinary optimisation processes within analysis-driven design processes. In this context, ontologies may serve as part of a reusable framework in which reasoning about artefacts, related processes and simulation tasks and their results is exploited to design, enact and adapt product development processes. Although not explored in this paper, the integration of different models and disciplines may also prove useful to guide designers and engineers in navigating analysis results and in exploring design alternatives.

We illustrate how meta models of processes and engineering models may be expressed in STEP/EXPRESS [10], a knowledge representation standard well-established in the automotive and other engineering domains, and how declarative mappings between meta models facilitate automated synthesis and adaption of complex processes. While no single language is likely to satisfy all use cases, the integration of models at the *meta-level* can be done using a small set of powerful languages. In this paper, we advocate the use of EXPRESS to formalise the necessary meta models along with model transformations. It has been shown that this language is sufficiently powerful to express and execute model mappings within a unified framework [16].

In Section 2, the role of ontologies in model-driven engineering processes is discussed. Section 3 presents the architecture of our framework and Section 4 discusses our approach to overcome disparities between different models. Section 5 elaborates on possible benefits of our framework and outlines future research directions.

[†]This work was partially funded by the Cooperative Research Centre for Advanced Automotive Technology under Project C4–801. We are grateful to Chris Seeling (VPAC) and Daniel Belton (General Motors, Holden Innovation) for providing a test-bed and domain expertise.

2. Design Optimisation Process

The desire to reduce costs and product cycles and the requirement to cope with increasing product complexity and stringent legal requirements have led to a paradigm shift towards virtualisation [8]. Early adoption of standardised modelling and simulation techniques and frequent model interchange strive to eliminate the need for physical artefacts and allow to overcome the aforementioned problems.

Multidisciplinary Design Optimisation (MDO) [5] is a form of virtual development where rigorous modelling and optimisation techniques are applied early in the design process to assess different design alternatives. Conflicting goals may require negotiation between teams to reach acceptable tradeoffs. A prerequisite for this is the representation of MDO *processes and information flow* in a way that permits semantic analysis. For this we have to focus on two views: the traditional product/design modelling view, and the explicit modelling of the process view. To combine these two aspects, we use what we call an Ontology-based approach, where "ontology" is used in the interpretation of [11] as meaning a set of concepts plus logical axioms that describe their interrelations.

Ontologies provide the means to represent the relationships between artefacts, their properties, and annotations made by designers/engineers in a way that enables semantic analysis and translation. To complement the artefact representation, a representation of the processes involved in the design is desirable. A consistent formal process model allows to connect engineering decisions and artefacts to the processes that induced them, and facilitates analysis to detect and resolve inefficiencies and change management. Here, extensions to well-known flow models [1] are necessary to adequately represent complex engineering tasks and models.

The approach to design support sketched in this paper heavily relies on formal ontologies to represent processes and design knowledge, as well as automated reasoning techniques to build, validate, and translate design processes and design knowledge. However, an evaluation of well-known languages and ontologies showed that no single ontology or language is sufficient to represent all desired aspects [15, 6]. Hence, different representations must be combined into a unified framework to overcome representational differences. The following sections introduce meta-modelling techniques and ontology mappings to facilitate the necessary transformations of processes, instances and data formats.

3. Ontology Architecture

The explicit use of *adaptors* [2] has been advocated to bridge gaps between ontologies, including for parametric configuration problems [7]. Although not based on adaptors, [12] show that domain-specific ontologies representing the function of devices can be related via a common model to extend reasoning about roles and functions beyond a single domain.

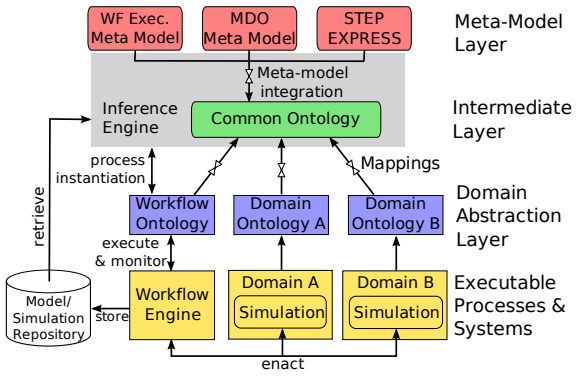


Figure 1: Framework Architecture

Our framework relies on domain experts and knowledge engineers to identify and represent relevant interactions between domain ontologies and formalise mappings between ontologies (cf. Figure 1).

We pursue a layered approach where meta models are located at the top, unified task and artefact ontologies comprise the intermediate layer, and domain-specific ontologies form the bottom layer in the ontology hierarchy. Concrete executable systems, such as CAD environments, optimisation tools, data bases and workflow orchestration engines, are located below the knowledge representation layers. The model and simulation repository acts as a conceptual "memory" component where ontologies, processes and information about (partially) executed simulations and their results are stored. The latter is obtained from the execution platform and accessed by the reasoning framework to draw inferences based on historic process information.

From analysis of individual domains, ontologies of domain-specific concepts, properties and relations are created. Process execution environments, for example workflow enactment systems, are treated in the same way. As a result, a set of domain ontologies is obtained. Domain-independent aspects and processes are found by generalisation of domain-specific ontologies to form the intermediate layer: By defining suitable ontology mappings, domain-specific knowledge is mapped into the unified common ontology at the intermediate level. Hence, it becomes possible to describe and reason about domain-independent and task-independent concepts, such as execution traces and execution histories. Common ontologies allow to *design*, *trace*, *reason about* and *execute* analysis-driven processes. Support environments developed for the design and execution of distributed scientific experiments have demonstrated that this is feasible in certain domains while hiding much of the underlying formal knowledge representation mechanisms from engineers [13].

Ontologies and inference systems that comprise the intermediate layer serve as a platform to integrate information and processes obtained from different domains and expressed in languages defined by different ontologies. This is critical when reasoning about processes and artefact-related infor-

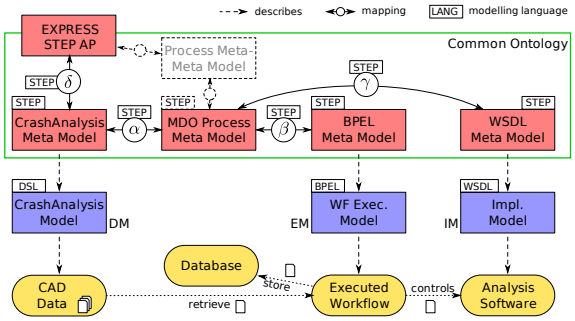


Figure 2: Ontology Mappings

mation simultaneously, although the two are not necessarily expressed in the same formal language.

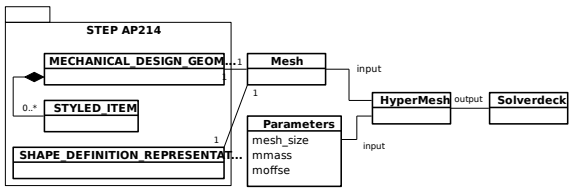
4. Ontology Integration

Translation between the intermediate layer and the ontologies below is accomplished by adaptors that map between domain-independent and domain-specific representations. The same idea is used to instantiate abstract process models at the intermediate level to generate concrete process specifications tailored to specific workflow execution environments, such as [13]. Differences between ontologies, workflow languages and data formats can be overcome at the meta model level [4].

Modelling Domain Specifics. Figure 2 illustrates an example in the context of crash impact analysis in the automotive domain: Geometry data and parameters of the analysis are stored in files as CAD and XML data, respectively. The structure of the files' contents and the steps involved in the analysis may be described by a domain-specific model, *DM*, expressed in a domain-specific modelling language. For brevity, we omit the models dealing with file formats and focus on the process aspect instead. The structure and semantics of the language are specified at the meta-level (cf. *CrashAnalysis Meta Model*). The model is represented in EXPRESS [10], which we have adopted as generic meta-modelling language. Available executable applications and their location are described in an *Implementation Model*, *IM*, which again conforms to its meta model. In this example we assume that applications are Web Service enabled and their location, input and output interfaces can be described using the Web Service Description Language (WSDL).

A simplified example of the Crash Meta Model describing our testbed is given in Figure 3a.[1] The model specifies the tasks involved in the analysis and their input and output data formats. For example, given a geometry model in IGES format and meshing parameters, task HyperMesh creates a mesh model represented as a Solverdeck (a tool-specific format). The details of how to invoke the local installation of HyperMesh are expressed in WSDL format (Figure 3b). Our MDO process meta model captures the essential process

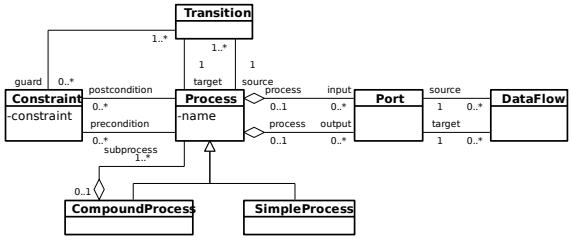
[1] We use UML notation in place of EXPRESS statements for brevity.



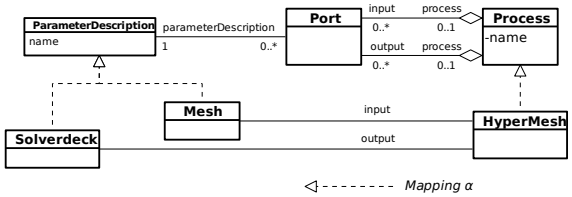
(a) Crash meta model

```
<definitions xmlns:s="..." > ...
  <message name="meshRequest">
    <part name="mesh" type="s:anyType"/>
    <part name="parameters" type="s:anyType"/>
  </message>
  <message name="meshResponse">
    <part name="solverdeck" type="s:anyType"/>
  </message>
  <portType name="mesherPort">
    <operation name="execute_mesh">
      <input message="tns:meshRequest"/>
      <output message="tns:meshResponse"/>
    </operation>
  </portType>
</definitions>
```

(b) WSDL (simplified)



(c) MDO Process Meta Model



(d) Mapping α

```
SCHEMA WSDLMapping;
  USE FROM ProcessModel; ...
  ENTITY MAP_PROCESS;
    process : Process;
    definitions : WSDL_Definitions;
  DERIVE
    match_op : LIST OF WSDL_Operation :=
     QUERY(o<*definitions.portType[1].operation
           |o.name = process.name);
  WHERE
    match_all:(SIZEOF(match_op)=1) AND
     MATCH_PORTS(match_op[1].input,process.input)AND
     MATCH_PORTS(match_op[1].output,process.output);
  END_ENTITY;
END_SCHEMA;
```

(e) Mapping γ

Figure 3: Meta models and mappings (simplified)

information in a language- and implementation agnostic representation [15] (Figure 3c). Tasks are modelled as entities, where control and data flow between tasks are captured in a graph structure. For each tasks, input and output data are represented as ParameterDescriptions that are annotated with semantic information in terms of STEP APs. Similarly, a model of BPEL specifications describing executable processes can be constructed [17].

Meta Model Driven Integration. In addition to meta models of input data and applications, ontologies describing critical concepts of MDO processes (*MDO Process Meta Model*) and workflow execution platforms (*BPEL Meta Model)* are assumed to be available. To execute the analysis specified in *DM*, it is necessary to create a model, *EM*, for the execution platform based on *DM* and *IM*.

Since the three domain models are expressed in different languages, these differences must be reconciled through model-integration at the meta-level. Rather than specifying mappings between each pair of meta models, we leverage our generic process model [15] to facilitate as an intermediary in the translation. Hence, mappings α, β and γ must be given that relate the three meta models: α maps the domain-specific meta model into our process model, and β formalises the translation from processes to BPEL; γ relates the WSDL ontology to equivalent generic entities in our process model. Figures 3d and 3e depict parts of such mappings.

STEP/EXPRESS plays a distinguished role in our framework, as its extensible structure allows to utilise the STEP Application Protocols (AP) to serve as entities on the domain- and meta-level, while the same language is also applied to describe mappings between the elements. For example, mapping δ indicates that the CrashAnalysis Meta Model is in part derived by extension of STEP APs. Therefore, established parts of the STEP Ontology can be incorporated into our framework. For example, analysis input and results can be linked to terms of AP 209 by mapping δ, hence permitting to reuse ontologies and detailed structure in the semantic annotations attached to tasks and design parameters in our MDO meta model.

Workflow Execution. The execution model *EM* can be synthesised automatically by applying mappings to the source model *DM*. Once the *EM* is obtained, the workflow platform executes the process by scheduling tasks to fetch and process the relevant data. Provenance information and other meta-data is recorded in a model repository. For brevity, ontologies and transformations related to data formats and units, data sources and the model repository have been omitted.

Adding Meta Layers. In limited domains it may be possible to carry the meta modelling idea even further by utilising *meta-meta models* describing essential properties of a domain to automatically infer certain mappings between ontologies at the meta level. For example, rather than specifying the mapping between BPEL and our process representation, it is possible to partially infer mappings by defining basic properties of processes, such as sequential and concurrent execution, or control and data flow. By mapping each ontology onto this generic model, equivalent representations for constructs in either input ontology can be synthesised in the other ontology.

5. Outlook on Supporting Optimisation

Optimisation-driven design processes have become prevalent in many disciplines. However, support to effectively *design* and *use* simulation processes has not been addressed satisfactorily. Our work on integrating representations of design artefacts, design processes, and process execution aims to address these challenges. The proposed architecture facilitates the support of a variety of different modelling and engineering tasks. For example:

Model management. Through model management enabled by our framework, each analysis can be based on a consistent and up-to-date global view of the design process.

Model provenance. Tracing the evolution of designs and analysis results becomes possible. In particular, the origin and assumptions underlying critical models and parameters can be recorded and later exploited to guide analysis efforts. For example, if it is known that a design constraint on a particular artefact part is only a default value, it may be opted to re-negotiate the exact value rather than spending much effort on satisfying the constraint (which may become obsolete in later design stages).

Simulation preparation. The experiment preparation stage *before* optimisation processes are executed may also benefit from rigorous modelling. For example, formal process descriptions can be used to ensure that a suite of experiments leads to compatible results that can subsequently be aggregated into a global view. Potentially inappropriate process inputs may be flagged by comparing parameter settings chosen by engineers with formal models of successful and failed executions stored in the repository.

Process adaption. System support for selection, composition and reuse of analysis processes may help to avoid errors in the setup stages and to shorten turnaround time. Flexible integration of different systems through automatic workflow synthesis and orchestration lessens the burden of integrating disparate systems from the IT personnel's point of view.

Execution. Using automated reasoning technology, process models can automatically be translated into workflow enactment models that are subsequently executed. This allows to automatically monitor, store, and reason about process outcomes that are handled by the MDO environment. Through common ontology and adaptors, simulation inputs and results can be compared. The flexibility offered by declarative ontology mappings enables to separate modelling and reasoning aspects from implementation-dependent aspects prescribed by different execution environments, such as scientific Grid environments and legacy applications.

Error handling. Simulations may also benefit from improved robustness of models and execution through semi-automated error recovery. If a simulation aborts due to modelling errors or out-of-range input values, formal ontologies and a repository of models and execution traces can help to determine whether a different model is available that does not exhibit the same problem and has been successfully applied using parameter values that match the current situation. Since permissible input values may not be known explicitly for complex models, case-based or machine learning approaches may be adopted.

Simulation reuse. MDO optimisation tasks can be adapted and streamlined if suitable results are available from previous similar analysis. Reusing the result of a simulation rather than performing redundant analysis becomes possible if it can be shown that the candidate result is an acceptable replacement in context of the current simulation.

Knowledge engineering. To curb the complexity underlying the creation of models, a hierarchy of meta models is proposed, where domain- and tool-specific models at the lower layers are reconciled into higher-level generic models that allow to translate between semantically overlapping domains and to mediate between heterogeneous infrastructure components. This approach allows to concentrate on conceptual modelling, without consideration of implementation-specific details, such as which database system to use.

The initial setup of the ontology mappings implies considerable effort, but we expect that the potential benefits outweigh the costs. By relying on existing engineering standards, such as the STEP application protocols and other XML data representations, the modelling effort can be directed at ontology mappings at the meta model layer, rather than on the lower-level ontologies and data format conversion implementation. Scenario-independent models and mappings that are developed for a given analysis scenario may be reused in the construction of models describing similar processes. In particular, mappings mediating between different data formats and execution platforms are likely candidates for reuse. Furthermore, the transition from current practices to the ontology-enriched process may be carried out incrementally to control the complexity of system migration.

Further work includes to extend formal mappings between ontologies to complex (common and domain-specific) entities and relations, as well as to investigate the application of different inference systems to support engineers in defining, maintaining and applying such transformations. Currently, we explore these technologies for consistency assessment of processes and their instances, as well as translation of higher-level representations into process representations that are used by different workflow execution engines for Grid environments. Detailed evaluation of possible inferences, impact on engineering practices and scalability assessment also remain for further investigation.

References

[1] C. Bock and M. Grueninger. PSL: A semantic domain for flow models. *Software Systems Modeling*, pages 209–231, 2005.

[2] B. Chandrasekaran, J. Josephson, and R. Benjamins. The ontology of tasks and methods. In *Knowledge Acquisition Modeling and Management Workshop*, 1998.

[3] C. Dartigues and P. Ghodous. Product data exchange using ontologies. In J. S. Gero, editor, *Proceedings of International Conference on Artificial Intelligence in Design (AID'02)*, pages 617–637, 2002.

[4] L. Deshayes, S. Foufou, and M. Gruninger. An ontology architecture for standards integration and conformance in manufacturing. In *International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, 2006.

[5] F. Duddeck. Multidisciplinary optimization of car bodies. *Journal of Structural and Multidisciplinary Optimization*, 34(1):375–389, 2007.

[6] A. Felfernig et al. Configuration knowledge representation for semantic web applications. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 17(1):31–50, 2003.

[7] D. Fensel, E. Motta, S. Decker, and Z. Zdráhal. Using ontologies for defining tasks, problem-solving methods and their mappings. In *EKAW*, volume 1319 of *LNCS*, pages 113–128. Springer-Verlag, 1997.

[8] N. Figay. Collaborative product development: EADS pilot based on ATHENA results. In P. Ghodus, R. Dieng-Kuntz, and G. Loureiro, editors, *International Conference on Concurrent Engineering*, pages 10–21. IOS Press, 2006.

[9] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer-Verlag, 2004.

[10] International Organization for Standardization. *10303-11:1994: Part 11: The EXPRESS language reference manual*. International Organization for Standardization, 1994.

[11] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.

[12] Y. Kitamura, Y. Koji, and R. Mizoguchi. An ontological model of device function: industrial deployment and lessons learned. *Applied Ontology*, 1(3–4):237–262, 2006.

[13] B. Ludäscher et al. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[14] A. Maier, H.-P. Schnurr, and Y. Sure. Ontology-based information integration in the automotive industry. In *International Semantic Web Conference*, volume 2870 of *LNCS*, pages 897–912. Springer, 2003.

[15] F. Maier, W. Mayer, M. Stumptner, and A. Mühlenfeld. Ontology-based process modelling for design optimisation support. In *International Conference on Design Computing and Cognition*, 2008.

[16] A. Plantec and V. Ribaud. PLATYPUS : A STEP-based integration framework. In *Interdisciplinary Information Management Talks (IDIMT-2006)*, pages 261–274, 2006.

[17] T. J. Reiter. Transformation of web service specification languages into UML activity diagrams. Master's thesis, University of South Australia, 2005.

[18] F. Tao, L. Chen, S. J. Cox, N. R. Shadbolt, C. Puleston, and C. Goble. Semantic support for grid-enabled design search in engineering. In *Semantic Grid Workshop, GGF9*, 2003.