Ontology-Based Process Modelling for Design Optimisation Support[1]

Franz Maier, Wolfgang Mayer, Markus Stumptner, Arndt Muehlenfeld
University of South Australia, Australia

While design processes in certain domains have shifted towards early adoption of simulation and virtualisation techniques, the integration and reuse of simulation and process information is not well-integrated into current development practices. We introduce a framework to integrate Multidisciplinary Design Optimisation (MDO) processes using ontological engineering. Based on a multi-disciplinary design scenario drawn from the automotive industry, we illustrate how semantic integration of process, artifact and simulation models can contribute to more effective optimisation-driven development. Ontology standards are evaluated to assess where existing work may be applicable and which aspects of MDO processes require further extensions.

Introduction

In the design and engineering context, ontologies provide an explicit formalisation of design knowledge that is otherwise distributed among several design teams [12]. Ontologies also aid in semantic interoperability between design disciplines due to the introduction of meta-models that serve as a linking element between disciplines [18], providing means to reason about process-, simulation- and domain-specific aspects [5]. The work described in this paper extends the emphasis from artifact-centric approaches to *process-oriented* ontologies to provide a comprehensive unified framework.

As designs become more complex, designers and engineers increasingly rely on tool support to manage not only design artifacts, but also the design

[1] This work was funded by the Cooperative Research Centre for Advanced Automotive Technology (AutoCRC), *Project 10: Integrated Design Environment*.

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. Similar issues arise where design activities are carried out in a distributed scenario [4].

Enacting, monitoring and predicting the execution of complex collaborative processes is crucial to react timely to changes and delays in subprocesses. This may span the entire process hierarchy from top-level business processes down to the execution of simulation tasks on cluster computers. Adequate process models allow to formalise and record the negotiation processes between design teams, as well as assessing their implications across organisational boundaries. Furthermore, process driven upper-level activities can be linked to data-centric optimisation tasks to automatically compose, enact and monitor the actual execution of optimisation tasks.

In this paper we outline an ontological representation of typical multidisciplinary optimisation (MDO) processes within analysis-driven design processes. Our ontological models comprise semantic descriptions of process constraints, design variables and further parameters originating from different design disciplines. We show how ontologies may serve as a reusable framework for MDO processes and how integrated reasoning about artifacts, related processes and optimisation tasks and their results may lead to more effective product development processes. We are not primarily concerned with the interoperability between engineering applications, but focus on how existing analysis tool chains can be employed more effectively.

In Section *Design Process Overview*, we present a general note on a framework for representation of optimisation processes and optimisation tasks at a domain-independent level. In Section *MDO Process Meta-model Development*, vital aspects of design process models are discussed and a meta-model of optimisation-driven design processes is introduced. In Section *Evaluation of Languages and Ontologies*, ontology languages are evaluated w.r.t. our modelling framework with focus on representation of and reasoning about processes. The architecture of our ontological framework for design optimisation processes is presented in Section *Task and Domain Representation*. Finally, related work is discussed, and our contributions and future work are summarised.

Design Process Overview

The design of complex products and systems is rarely carried out in a single monolithic step, but is broken down into a number of stages. The design cycle is structured in different activities, where initial tasks are concerned with exploration of design goals and alternatives at a conceptual level; detailed exploration of selected design alternatives and design artifacts is done only later in the product development cycle.

Tasks allocated to each state are typically distributed among a number of teams, each concerned with a particular aspect of the design under consideration (see Fig.1). Stages of the design process are synchronised by milestones, where the outcomes of the previous design activities obtained from different teams are assessed, integrated and approved as input for the subsequent stage. If a design does not satisfy certain required constraints, negotiation between design teams takes place in order to revise the specifications.

For example, the initial design activities in the automotive industry are concerned with market analysis, styling and overall architectural decisions; detailed manufacturing assessment and construction, integration and testing of physical prototypes are done only later in the process. Since the design of a car's underbody and its engine block may be carried out by different teams in parallel, teams must ensure that the interfaces of engine block and underbody remain compatible (dimensions, mount points, etc).
While widely employed, the approach is not free of drawbacks in practice:

- Infrequent information interchange only at milestones may lead to inefficient development, where different teams aim at meeting specifications that have become obsolete due to changes made elsewhere.
- Semantic integration of results stemming from different disciplines may be challenging due to varying level of detail, data representation and terminological mismatches. Incompatible data formats and tools may also contribute to poor data quality.
- Assessment and integration of information at a milestone may be difficult, since design activities may be carried out at different levels of detail or may not reach the same level of maturity at a synchronisation point.

Multidisciplinary Design Optimisation (MDO)

Multidisciplinary Design Optimisation (MDO) is a form of virtual development where rigorous modelling and optimisation techniques are applied starting early in the design process, to obtain a coarse understanding of different aspects of a design across a number of heterogeneous domains. Rather than optimising each discipline separately, all disciplines are analysed in parallel and the results are merged with the intent to obtain the best design alternative as a compromise of all included disciplines.

The virtual design artifact is abstracted into a number of design variables that represent relevant properties for each domain under consideration. Relationships, such as trends and correlations, between design variables derived from domain-specific models and simulations express properties of the design and guide designers in selecting and refining design alternatives. Typically, these relationships are not explicitly known and must be obtained by simulation.

The Design of Experiment (DoE), that is, selecting appropriate parameters for simulation, is a challenging problem given the increased number of design variables and interactions between different domains. Since a single experiment may run for many days even on high-performance computing environments, careful choice of parameter values is critical.

A related problem is the representation of the *Design Interfaces* that specify the design variables. All solutions of sub-designs connected to the interface must assign compatible values. *Design Constraints* specify partial value assignments to variables of design interfaces and determine the design space that remains for investigation. Through this representation, both goals and (partial) design solutions are communicated between otherwise independent design teams in the optimisation processes. For example, a design interface between engine block and car body could specify that both components must agree on the location, shape and strength of joints. A particular design solution would assign concrete values to the interface variables.

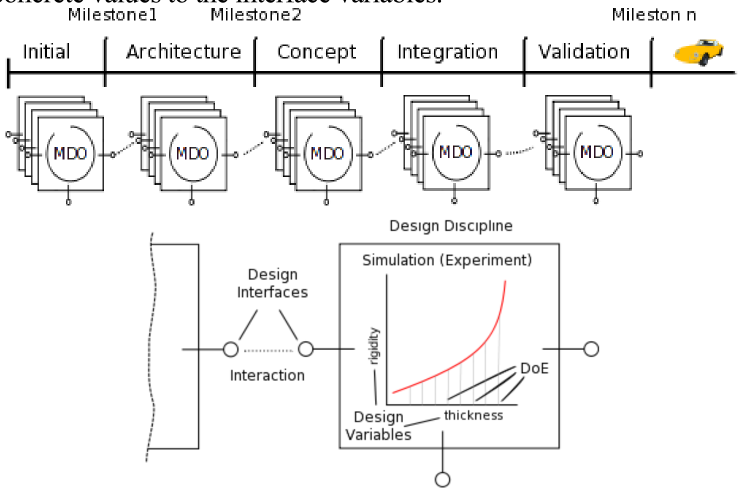

Fig. 1. MDO-driven design process

Since the representation of goals, design constraints and design solutions may vary, semantic-preserving translation is critical to ensure a consistent overall design: (i) Multiple interfaces that represent a design artifact at different granularity can coexist in a design problem. For example, there could be an abstract interface representing an engine block and lower level interfaces representing the individual components comprising the aggregate component. Since the representation and level of abstraction may differ, semantic-preserving translations between the two representations must be performed. Design goals for the aggregate component are mapped to sub-goals, while detailed simulation results obtained for each component must be aggregated into a consistent result at the more abstract level. (ii) Similarly, the analysis granularity may vary among components. To ensure a consistent model, abstract and detailed representations of design constraints and simulation results must be mapped onto each other.

A design optimum obtained from an MDO process execution may affect other DoEs or their optimal solution, which may require negotiation between teams to reach acceptable tradeoffs, subject to the list of issues from the previous section. 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 theses two aspects, we use what we call an Ontology-based approach, where “ontology” (a term used in many meanings in the literature) is used in the interpretation of [16] as meaning a set of concepts plus logical axioms that describe their interrelations. The requirements of the domain drive the requirements for the axioms and the language in which they are expressed.

Ontology development for MDO

Provided with above understanding of MDO processes, a number of issues can be addressed by introducing an appropriate level of formalisation, in particular, the ability to merge, reuse, and locate past MDO runs; provision of interoperability between different MDO domains and abstraction levels; design and execution tracking; specification of processes, products, and design objectives; visualisation of MDO results; support of the negotiation process; data translation (from and to CAD and solver systems); administration of constraints; representation of design interfaces, constraints, process elements, simulation results (inputs, outputs, data formats); and (most ambitiously) synthesising and monitoring of process executions. These criteria will determine the selection of representation choices.

MDO Process Meta-model Development

System Philosophy

To address the research questions posed in the previous section, two aspects of design and optimization processes must be considered: the *process aspect* and the *artifact representation*.

Process representation

The process representation is concerned with the representation and execution of design and engineering processes within and across organisational boundaries. A consistent formal process model allows to connect design decisions and artifacts to the processes that induced them. This is important in particular for process analysis to detect and resolve inefficiencies, as well as to track the evolution of processes over time. This can be considered an extension to the well-known “Corporate Memory” idea advocated, for example in [10], where design artifacts are stored to be retrieved by designers for later reuse. Having process information attached allows to extend this idea to the entire product development and deployment life cycle, such that it becomes possible to query process-related properties.

Common to all scenarios mentioned here is the requirement for a process model to represent not only the flow between process activities, but also the preconditions, effects and inter-dependencies between sub-processes at different levels of the hierarchy. While flow-models for processes have been extensively researched [1], the formalisation of semantic representations of activities in design and engineering processes has not been fully addressed so far.

Artifact representation

Adequate representations of structure, function and semantic annotation of design artifacts are essential requirements for reasoning about design artifacts as well as design processes, their prerequisites and their results. Ontologies provide the means to represent the relationships between artifacts and sub-artifacts, as well as (material and domain-specific) properties and annotations made by designers/engineers, in a way that is amenable to semantic analysis and translation. This aspect is not new; in fact there is much work in this area that we incorporate into our approach. Ontologies designed to express hierarchical and functional rather than flow information are required to express complex artifacts and related constraints. Standards like STEP [14] and ontologies to represent function [17] serve as a starting point for the specific purpose of artifact modelling.

Example Application

The optimisation problem outlined in Figure 2 and Table 1 serve as a running example in this paper. The problem consists of a prismatic sheet metal structure that serves as a stub for a model of a real vehicle [23]. The model is parametric such that different combinations of artifact properties like material or number and location of structural elements may be investigated. Aspects of the MDO process itself, for example analysis granularity, are also configurable.

Table 1 Configuration of the Benchmark Example

Parameters	Values	Parameters	Values
Nodes (crash model)	ca. 6000	Design concepts	1
Volume elements (crash model)	ca. 20	Objectives	1
Independent design variables	16	Simulations	200
Design disciplines	1	Duration of optimisation	ca. 72h

The analysis problem under consideration is a crash simulation where the prism is equipped with a mass element and is driven into a fixed wall. The energy absorption of the model at different points in response to changes to structural and material properties is obtained as output. Crashworthiness is considered a good test bed for MDO processes since a considerable number of design variables are involved and the problem is one of the central tasks in virtually every vehicle design process.

The workflow shown in Figure 2 describes part of the basic structure of the problem. It consists of four steps: *configuration*, *gambit*, *mesh* and *lsdyna*. The four sub-processes represent the setup process, the geometry generation process, the meshing process (that is, creating the finite element mesh from a geometric model), and the solving process (simulation of finite-element model), respectively. To allow engineers to experiment with parameters, sub-processes may be configured to reuse a previous result rather than recompute an analysis.

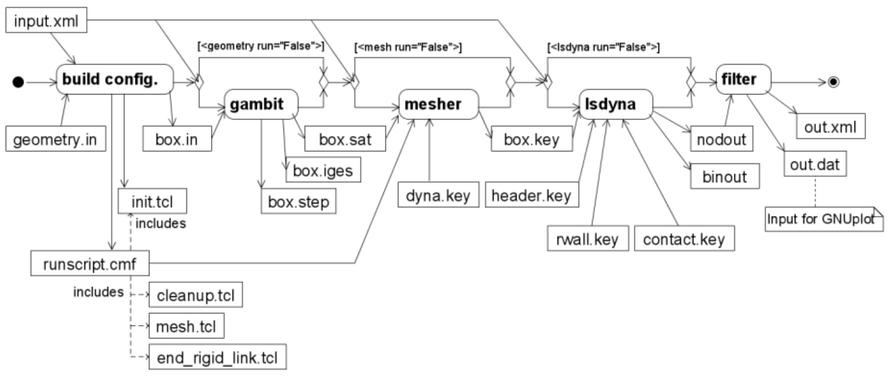

Fig. 2. Sample MDO Workflow

A Process Meta-Model for MDO

As the basis for the joint semantic representation a generic meta-model for optimization-driven design processes has been created (see Figure 3). We use the Unified Modeling Language (UML) due to its widespread usage.

The central model component of the model is *Process*, which represents generic activities in a process. Specialisations for trivial tasks, *SimpleProcess*, and for hierarchically structured sub-processes, *CompoundProcess*, exist. Each process has *Ports* that represent inputs required and results generated by each process. While ports of *SimpleProcess*es are determined by the represented activity, *CompoundProcess*es aggregate the inputs and outputs of its constituents.

Each *Port* is associated with a *ParameterDescription* that acts as a placeholder for the semantic specification of the data that is obtained from/generated at a port. Through references to *ModelVariable*s, a parameter description may refer to aspects of a particular MDO problem. Similar to [22], the separation of *ParameterDescription* from *Port* allows to encapsulate language and implementation of ontologies within a separate entity and avoids duplication. Most closely related to our notion of *ParameterDescription* is the concept of typed input and output channels that are integral part of well-known Web service and workflow models [21].

Processes are connected through source and a target *Transition*s, where each transition is guarded by a *Constraint*. Constraints are an abstraction of logical expressions over instances of *Port* and *ParameterDescription*s. Control flow between processes along transitions is guarded by constraints that determine whether a transition may be followed or if the transition is blocked. This model is generic in that it subsumes other well-known process languages such as BPEL and UML Activity Diagrams. Constraints also express preconditions and effects of a particular process step, represented as expression over ports and parameters.

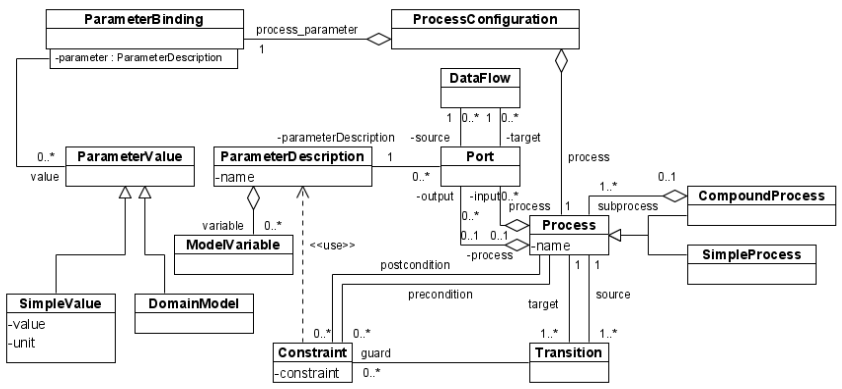

Fig. 3. MDO Meta-Model

Entity DataFlow represents an abstraction of “data channels” that determine the flow of values, design artifacts and other MDO-related data between ports.

Entity *ParameterValue* represents data that is passed into or generated by process activities. This includes complex structures such as entire geometry models or response surfaces generated from domain-specific analyses. Again, this entity serves to separate domain-specific and process-specific ontologies and representations.

The meta-model described here is an abstraction of concrete process models used in a particular domain, which in turn are specifications of possible execution scenarios that may occur. As such, the meta-model must be instantiated for a particular domain to obtain domain-specific process models. As discussed in Section *Evaluation of Languages and Ontologies*, we have investigated the translation of the generic UML model into more formal representation languages to allow automated reasoning and synthesis of instantiated models. The model as depicted omits environment specific implementation details, such as particular workflow enactment engines, which must be added in the course of refinement. We have equipped our meta-model with mechanisms to represent different versions, configurations and implementation platforms, similar to what is commonly referred to *grounding* in Web service specification formalisms. Similarly, ontologies and languages used to represent *ParameterDescriptions* and *Constraints* have been deliberately left unspecified in our meta-model to allow different (domain-specific) ontology languages to interface with our process model. This way, different formalisms may be used to describe processes and design artifacts. As shown in the following section, no single formalism is perfectly suitable for the purpose and we work on the basis of embedding these different perspectives into a common framework.

Evaluation of Languages and Ontologies

We evaluate whether a language satisfies the following requirements: existence of automated reasoning capabilities; support for different inference strategies; support for quantification over attributes and anonymous concepts; availability of robust implementations and tool support for knowledge engineers; whether implementations scale to non-trivial problems. In the following, we briefly introduce and evaluate well-known representation formalisms and ontologies.

Representation Formalisms

Table 2 summarises our analysis of representation languages. For space reasons, only the most promising formalisms are considered.

Table 2 Characteristics of Representation Languages ('-' not available, '+/-' available to some degree, '+' available)

Requirement	**PSL**	**F-Logic**	**KIF+CL**	**STEP**	**OWL**	**RDF(S)**	**XML**
Concepts	+	+	+	+	+	-	+
Attributes	+	+	+	+	+	+	+
n-ary Relations	+	+/-	+	+	+	+	+
Functions	+	+/-	+	+	-	-	-
Instances	+	+	+	+	+	+	+
Rules	+	+	+	+	-	-	-
Formal Semantics	+	+	+	-	+	+	-
Inferences							
Subsumpt. checking	+	+	+	+	+	+	-
Constraint checking	+	+	+	+	+	-	-
Concept hierarchies	+	+	+	+	+	+	-
Subtyping	+	+	+	+	+	+	-
Subsumption	+	+	+	-	+	-	-
Complex Constr.	+	+	+	+	-	-	-
Domain-spec. prop.	+	+	+	+	+	+	-
Modularity	+/-	+	+	-	-	+/-	+

UML lacks standardised semantics and is thus unsuitable for reasoning. The Object Constraint Language (OCL) provides means to express constraints over sets of objects that cannot be expressed by a diagrammatic notation.

KIF and CL represent an interchange format for logic languages and enable translation between ontologies. F-Logic, which has similar characteristics, has been used in design environments [18].

Constraint satisfaction techniques are an efficient mechanism to search for solutions to logic theories and have successfully been applied to industrial problems including design environments, serving as the standard representation form for configuration and process modelling tasks [7], [24].

Description Logics (DL) and related Semantic Web standards OWL and DAML+OIL are less expressive than full First Order Logic. They have strong reasoning and tool support but lack core features for the modelling of product entities as, e. g., the representation of resources [7]. OWL does not provide support for modularity and complex constraints, two pivotal criteria in design environments. XML lacks formal semantics and axioms; hence, reasoning capabilities are not available.

Ontologies

Table 3 summarises our analysis of ontologies. The Standard for the Exchange of Product Model Data (STEP) [14] tailored for a comprehensive representation of product data and therefore can be employed in a manufacturing environment to capture artifact-specific, geometric and domain-specific data. On a meta-model layer STEP can serve as a representation of an artifact ontology. STEP tools, such as translators between different CAD-applications, exist and are deployed in design environments.

Table 3 Characteristics of Ontologies ('n.e.' not evaluated)

Requirement	**PSL**	**STEP**	**CPM**	**OAM**	**FBS**
Processes	+	-	-	-	+
timed	+	-	-	-	-
concurrent	+	-	-	-	-
exceptions	-	-	-	-	+
asynchronous	+	-	-	-	-
events	+	-	-	-	+
Intention Recording	+	-	+/-	+	+
Robustness	+/-	+	n.e.	n.e.	n.e.
Scalability	+	+	n.e.	n.e.	n.e.
Industry Acceptance	+/-	+	-	-	n.e.
Execution engine	+/-	+	-	-	n.e.
Tool support	+/-	+	-	+/-	n.e.

The Core Product Model (CPM) covers engineering information shared in product development [22]. The model intends to capture generic product information and structure. The Open Assembly Model (OAM) extends CPM and represents a model and exchange protocol for assembly. Both do not support process modelling.

The function-behaviour-structure (FBS) ontology [11] represents a framework to classify processes that supports the situated design of processes. It has been used in a number of design-support tools, but must be complemented by a more detailed representation to capture detailed process-related information. The Process Specification Language (PSL) was designed to describe process entities using a formal approach based on first-order logic [1] to describe and reason about manufacturing processes. PSL meets most of the evaluation criteria, but has no procedures and limited support for modular reasoning.

Kitamura et al. [17] provide an ontological modelling framework of functional knowledge including a controlled vocabulary and an ontology of device and function. The framework, further discussed in section *Related Work* is able to represent design rationale and provides a hierarchic decomposition of functions.

Analysis and Recommendation for MDO

The analysis in previous sections shows that PSL, F-Logic, KIF and CL fulfil most evaluation criteria. We utilise PSL as a formal basis to reason about task ontologies and execution traces, but specialise the ontology to aspects specific to MDO.

While PSL is well-suited for representing processes, detailed artifact- and analysis-related knowledge cannot be captured. To bridge the gap, complementary ontologies must be applied. Here, STEP is a suitable candidate, since standardised data formats as well as mappings to formal frameworks are available. STEP/EXPRESS allow to capture and analyse structural properties of artifact models and concrete instances. The absence of formal semantics can be resolved by mapping to a FOL-based formalism. This approach has already been used for the optimisation of diagnostic processes in an industrial environment [25].

Task and Domain Representation

Task and Domain Ontologies for MDO

The meta-model described in Section *A Process Meta-Model for MDO* represents a generic framework that describes model-driven design processes at an abstract level. While at this level generic concepts such as design variables, control flow, tracing and grounding of executions are introduced, the model must be specialised to a particular domain and organisational structure to be applied in practice.

This specialisation may be interpreted as ontology engineering problem, where a generic ontology is adapted to suit particular domain- or task-specific aspects [20]. The adaption can be approached in two separate dimensions to yield a *task ontology* and a *domain ontology*. Task ontologies represent knowledge that is particular to performing individual tasks (from the process point of view) at a domain-independent level, while domain ontologies capture the conceptual primitives necessary to describe a particular domain. This separation of concerns allows to represent knowledge about individual domains separately from knowledge about how domain knowledge is manipulated [2].

Task ontologies specialise the generic meta-model to express ways to approach a design or optimisation problem and their relationships. Hence, task ontologies provide the means to reason about and link a given set of *goals* to processes that achieve these goals.

Domain ontologies on the other hand focus on the representation of design artifact and domain-specific information. Established standardised representations such as STEP [14], the Core Product Model (CPM) and the Open Assembly Model (OAM) [22] may be applied and extended to serve as formal representations or meta-models for domain ontologies. Ontologies developed to describe the structure and function [17] may also be utilised within our framework to represent the function of artifact parts and elements that may be used as replacements.

The explicit use of *adaptors* [2] has been advocated to bridge gaps between ontologies, including for parametric configuration problems [8]. Although not based on adaptors, in [17] it is shown 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.

Framework Architecture

Our framework relies on domain experts and knowledge engineers to identify and represent relevant interactions between domain ontologies and formalise mappings (our manifestation of ``adaptors") between ontologies. Hence, it becomes possible to support designers and engineers in planning, revising, and executing and analysing MDO related processes and their results. The proposed approach is illustrated in Figure 4.

We pursue a layered approach where our meta-model is located at the top, task and artifact ontologies comprise the intermediate layer, and domain-specific ontologies form the bottom layer in the ontology hierarchy.

Concrete executable systems, such as CAD environments, MDO optimisation tools, data bases and workflow orchestration engines, are located in lower layers.

From analysis of individual domains, ontologies of domain-specific concepts, properties and relations are created, as well as specifications of domain-specific analysis primitives. Process execution environments, for example workflow enactment systems, are treated in the same way. As a result, a set of domain ontologies is obtained.

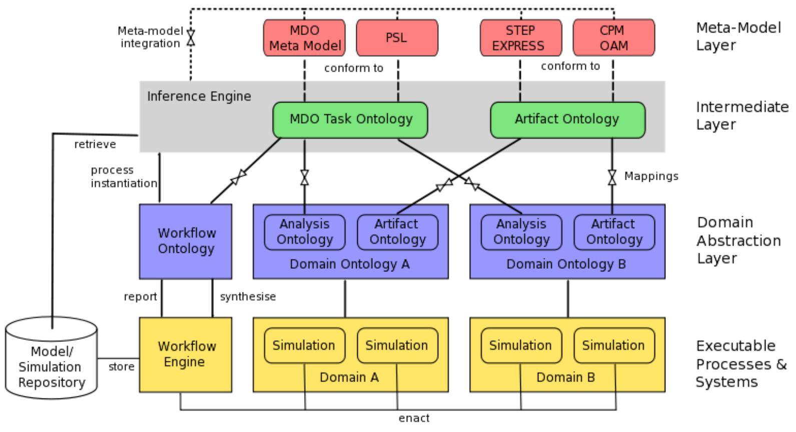

Fig. 4. Ontology Architecture

Domain-independent aspects and processes are found by generalisation of domain-specific ontologies to form the intermediate layer. By defining suitable ontology mappings, specific knowledge is mapped into the unified ontologies at the intermediate level. Established engineering practices and processes may also be incorporated into the MDO task ontology. Similarly, a generalised artifact ontology is constructed from the artifact domain ontologies.

Task and artifact ontologies at the intermediate layer must conform to the meta-models in the upper layer. This is desirable for two reasons: first, a common meta-model allows to describe and reason about domain-independent and task-independent concepts, such as execution traces and execution histories. The meta-model defined in Section *A Process Meta-Model for MDO* defines the necessary framework and concepts. Second, task and artifact ontologies need not be expressed in the same formal framework and may need to be reconciled to obtain an inference framework that can handle mixed expressions (see the following section).

Hence, 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. Common task and artifact ontologies allow to *design*, *trace*, *reason about* and *execute* analysis-driven processes in a language that is suitable for designers and engineers. Support environments developed for the design and execution of distributed scientific experiments have demonstrated that this is feasible without exposing the underlying formal knowledge representation mechanisms to designers and engineers [21]. Translation between the intermediate layer and the ontologies below is accomplished by adaptors that map between domain-independent and domain-specific representations.

The Benefits of Formal Ontologies for MDO

Compliance to the meta-models in the top layer allows use of automated reasoning systems to integrate ontologies and artifact ontology in a uniform framework [6].

We use PSL to formalise the task ontology, while the STEP/EXPRESS [13] framework is applied to define the artifact ontology. For space reasons we will briefly depict some representative examples of the automated process manipulation.

Process execution

Using automated reasoning technology, process models can automatically be translated into workflow enactment models that are executed. This allows to automatically track, store, and reason about process outcomes that are handled by the MDO environment. Through common task and artifact ontologies and adaptors, simulation inputs and results can be compared and possible changes to the process may be suggested and validated.

Simulation reuse

For example, an MDO optimisation task can be adapted and streamlined if suitable results are available from previous similar analysis. To re-use the result of a past 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. This requires reasoning about the execution history of past and current processes and to compare assumptions, results, constraints and goals of execution traces.
In our framework, abstract process specifications, optimisation tasks and traces of process executions are represented using the PSL ontology. For example, Figure 5 expresses that activity *lsdyna* is a subactivity of crashworthiness analysis that requires as input a finite-element model. Similar-

ly, execution traces of process instances are encoded as PSL statements (Figure 6). Each trace represents the actual execution of a process, where each node is associated with information about its execution (for example, machine node identifier and time stamps). Figure 6 represents a fragment of an execution of the workflow in Figure 2, where an execution of task *HMBatch* is immediately followed by an execution of *LsDyna*, and no other activity immediately follows *HMBatch*. Also given are execution-specific data, in this case time stamps and software version.

```
(activity Crashworthiness)
(exists (?mesh ?result)
  (and (activity LsDyna(?mesh,?result))
    (subactivity LsDyna(?mesh,?result) Crashworthiness)
    (forall (?occLD)
        (=> (occurrence_of ?occLD LsDyna(?mesh,?result))
            (and (occurrence-input ?mesh ?occLD)
                 (occurrence-output ?result ?occLD))))))
```

Fig. 5. Task Specification in PSL

```
(and (occurrence_of occHMB HMBatch(geom1,mesh1))
     (occurrence_of occLD  LsDyna(mesh1,result1))
     (occurrence_of occC   Crashworthiness)
     (= occLD (successor LsDyna(mesh1,result1) occHMB))
     (= (begin_of occHMB) '23/11/07 12:00')
     (= (end_of occHMB) '23/11/07 18:00')
     (holds (version 1.1) occHMB)
     (forall (?occ ?m ?r)
            (=> (= ?occ (successor LsDyna(?m,?r) occHMB))
             (= ?occ occLD))))
```

Fig. 6. Process Execution in PSL

```
ENTITY Mesh;
  model_of: Part;
  points : SET of Point;
  granularity : NUMBER;
END_ENTITY;
FUNCTION compatible(m1,m2:Mesh):Boolean;
  RETURN m1.granularity = m2.granularity;
END_FUNCTION;
```

Fig. 7. Mesh Compatibility Constraints

Assume that an analysis is to be conducted to compare the crash behaviour of two alternative designs. For two crash simulations of the same artifact part to be comparable, it is required that the finite element models used in both simulations are of equivalent granularity. Furthermore, due to algorithmic differences between software versions, it is required that both meshes were computed using equivalent software (version number). Figure 7 formalises this aspect of compatibility between mesh models in our artifact ontology.

The formal representation of compatibility requirements and execution traces allows to query and retrieve from the repository simulation results that are compatible with the requirements of the current experiment. In our example, requirements imposed on the input mesh model can first be transformed using the artifact ontologies to obtain constraints on mesh models restricting software version numbers. These requirements can then be injected into the models of each candidate execution trace to assess whether all process activities manipulating the model satisfy the constraints. Conforming traces can be presented as candidates, while a constraint violation would lead to the rejection of a candidate.

Simulation Selection

Given that processes and process executions are represented in the same framework, similar reasoning may be applied in the experiment preparation stage *before* optimisation processes are executed. For example, PSL 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.

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 invalid input values, formal ontologies and a repository of models and execution traces support determining whether a different model is available that does not exhibit the same problem and has been applied using parameter values matching the current situation.

Related Work

Ontologies to exchange product data are discussed in [5], focusing mostly on the interoperability aspect. However, translation and mediation remains an issue, as standards are often not fully implemented in practice. In [15] a framework is introduced where mediation is negotiated *dynamically*, based on a knowledge model rooted in the function-behaviour-structure ontology [11]. We assume that the engineering processes are already in place and the focus is therefore on supporting engineers in *using* the available resources more effectively.

In [19], extensions of STEP to represent analysis-driven design activities are investigated. Here, we extend the approach to integrate artifact and process models into a unified framework.

An ontology of device and function and a vocabulary has been implemented by [17]. The framework partially formalises the functionality of an artifact and covers parts of the intended artifact ontology and the functional model of a design artifact in our architecture.

In [9] Constraint-based mechanisms have been applied to validate a given design but process modeling and manipulation are not addressed.

Grid services are recently becoming a factor in design and scientific environments due to their ability of providing resources that substantially accelerate the processing of simulation tasks [21]. In [3] ontology engineering is applied to support the design and execution of scientific simulations. Different from our work, ontologies are used to compose workflows rather than to relate different processes and their executions to each other.

Conclusion

Analysis and optimisation-driven design processes have become prevalent in many disciplines. However, support for designers and engineers to effectively *use* the results of simulation processes has not been addressed satisfactorily. Our work on integrating representations of design artifacts, design processes, and process execution aims to address these challenges.

Based on formal ontologies representing design processes, simulation and optimisation tasks and abstractions of design artifacts, we provide a framework where prerequisites and results of different simulation tasks are related through common ontologies and synergies between analysis tasks may be exploited. Separation of task-related and artifact-specific representations allows to employ different standards to represent each. Mappings between common and domain-specific ontologies allow to interpret and reason about process executions and domain-specific simulation results on a meta-level.

We evaluated different ontologies and standards to assess their suitability to represent aspects relevant to design optimisation processes and design artifacts, respectively. As a result, PSL, STEP and OAM were identified as the ontologies that best fit into our generic framework, with [13,21] remaining candidates for future integration.

Further work includes extension to formal mappings between common and domain-specific ontologies, as well as to investigate the application of different inference systems. We investigate adapting constraint technologies to our ontological framework and contrast that approach with FOL-based theorem provers. Currently, we exploit these technologies for consistency assessment of processes and their instances. Integration of one of the publicly available workflow execution engines for Grid environments into our framework and detailed evaluation of possible inferences, impact on engineering practices and system scalability also remain for further investigation.

Acknowledgements

Domain knowledge and processes are developed jointly with the Victorian Partnership for Advanced Computing and General Motors/Holden Innovation. We thank Chris Seeling (VPAC) and Daniel Belton (GM/H) for the reference problem design and insights into the MDO process.

References

1. Bock C, Grueninger M (2005) PSL: A semantic domain for flow models. Software Systems Modeling: 209-231
2. Chandrasekaran B, Josephson J, Benjamins R (1998) The ontology of tasks and methods. Proc. of KAW'98
3. Chen L, Shadbolt R, Tao F (2005) Semantics-assisted problem solving on the semantic grid. Computational Intelligence 21(2): 157-176
4. Chira C, Roche T, Tormey D, Brennan A (2004) An ontological and agent-based approach to knowledge management within a distributed design environment. Proc. of DCC'04: 459-478
5. Dartigues C, Ghodous P (2002) Product data exchange using ontologies. Proc. of AID'02: 617-637.
6. Deshayes L, Foufou S, Grueninger M (2006) An ontology architecture for standards integration and conformance in manufacturing. Proc. of IDMME
7. Felfernig A, Friedrich G, Jannach D, Stumptner M, Zanker M (2003) Configuration knowledge representations for semantic web applications. AI EDAM 17(1): 31-50.
8. Fensel D, Motta E, Decker S, Zdráhal Z (1997) Using ontologies for defining tasks, problem-solving methods and their mappings. Proc. EKAW: 113-128
9. Fowler D et al. (2004) The designers' workbench: Using ontologies and constraints for configuration. Proc. AI2004/SGAI'04
10. Fruchter R, Demian P (2002) CoMem: Designing an interaction experience for reuse of rich contextual knowledge from a corporate memory. AI EDAM 16(3): 127-147
11. Gero JS, Kannengiesser U (2007) A function-behaviour-structure ontology of processes. AI EDAM 21(4): 379-391
12. Gómez-Pérez et al. (2004) Ontological Engineering. Springer
13. Haymaker J, Kunz JC, Suter B, Fischer MA (2004) Perspectors. Advanced Engineering Informatics 18(1): 49-67

14. ISO 10303-11 (1994) Ind. automation systems and integration-Product data representation and exchange-Part 11:The EXPRESS language ref. manual.
15. Kannengiesser U, Gero JS (2006) Towards mass customized interoperability. Computer Aided Design 38(8): 920-936
16. Kifer M, Lausen G, Wu J (1995) Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM 42(4): 741-843
17. Kitamura Y, Koji Y, Mizoguchi R (2006) An ontological model of device function: industrial deployment and lessons learned. Applied Ontology 1(3-4): 237-262
18. Maier A, Schnurr HP, Sure Y (2003) Ontology-based information integration in the automotive industry. Proc. Intl. Sem. Web Conf., LNCS 2870: 897-912
19. Maier F, Stumptner M (2007) Enhancements and ontological use of ISO-10303 (STEP) to support the exchange of parameterised product data models. Proc. ISDA'07: 433-440
20. Mizoguchi R, Vanwelkenhuysen J, Ikeda M (1995) Task ontology for reuse of problem solving knowledge. KB&KS: 46-59
21. Oinn T et al. (2006) Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10): 1067-1100
22. Rachuri S et al. (2005) Information models for product representation: core and assembly models. Int. Journal of Product Development 2(3): 207-235
23. Seeling C (2007) User Manual for the Crash-box MDO reference problem
24. Thiagarajan R, Stumptner M, Mayer W (2007) Semantic web service composition by consistency-based model refinement. Proc. 2nd IEEE Asia-Pacific Service Computing Conference (ASPSCC'07): 336-343
25. Wilmering T, Sheppard J (2007) Ontologies for data mining and knowledge discovery to support diagnostic maturation. Proc. 19th Intl. Workshop on Principles of Diagnosis (DX'07): 210-217