A Model Driven Framework for Integrated Computational Materials Engineering

Prasenjit Das
Tata Consultancy Services Limited
Kolkata, India
+91-33-66884653
prasenjit.d@tcs.com

Raghavendra Reddy Yeddula
Tata Consultancy Services Limited
TRDDC, Pune, India
+91-20-66086334
raghavendrareddy.y@tcs.com

Sreedhar Reddy
Tata Consultancy Services Limited
TRDDC, Pune, India
+91-20-66086302
sreedhar.reddy@tcs.com

ABSTRACT

Integrated computational materials engineering (ICME) is a new approach to the design and development of materials, manufacturing processes and products. The approach proposes using a combination of modeling and simulation, data driven reasoning and knowledge guided decision making to a) speed up the development of new materials and manufacturing processes, and b) enhance the quality and time-to-market of products by integrating material design with product design. However industrialization of this approach requires strong automation support. Modeling and simulation is a highly knowledge intensive activity and integrated design requires knowledge cutting across several design domains. For the industrialization vision to succeed, it is essential to capture this knowledge and make it available in a usable form for people not so skilled in these areas. With this motivation, we are building a comprehensive computational platform to support this emerging design paradigm. The platform is built on model driven engineering principles. We present some of the key ideas of the platform, discuss the modeling challenge involved and present the modeling framework we have developed to address this challenge. We also briefly discuss how model driven techniques have been employed to automate some of the key aspects.

Categories and Subject Descriptors

Computing methodologies → Modeling

General Terms

Design, Languages, Theory.

Keywords

ICME, Model-driven Architecture, Meta Modeling, Modeling Framework, Ontology.

1. INTRODUCTION

A material's properties such as its tensile strength, hardness, fatigue life, etc., are a result of its internal structure called microstructure. A material's microstructure depends not only on the chemical composition of the material but also on the processes it is subjected to. Materials engineers play with variations in chemical compositions, processes and process parameters in order to achieve required microstructure that gives rise to the desired properties. However these relationships are not well understood. As a result, a lot of trial and error and experimentation goes into designing a material. It takes anywhere between 10 to 20 years for a new material to find its way from research stage to industrial usage. Lack of integration between material design and product design is another problem. A product designer has limited visibility into the internal structure of the material and how that structure changes during a manufacturing process. Hence there is considerable uncertainty as to what final properties the material ends up with. To overcome this, product designers typically fall back on tried and tested materials and build in extra margin of safety into their designs.

There is a new design paradigm called integrated computational materials engineering (or ICME for short) [1, 2] that tries to address these issues through a computational design platform. ICME supports integrated design of materials, products and manufacturing processes. It uses modeling and simulation, knowledge guided decision making and data-driven reasoning for a systematic exploration of the design space. ICME is widely recognized as a paradigm changer that is expected to significantly reduce the dependence on trial and error based experimentation cycles. This is expected to result in a) faster development of new materials, and b) significant improvement in quality and time-to-market of products by integrating material design with product design. However, industrialization of this approach has many roadblocks to overcome [3]. Modeling and simulation is a highly knowledge intensive activity. Models exist at multiple length scales. In an integrated design, one has to worry about a multitude of phenomena. Choosing right models for these phenomena, at right scales, with right parameters, and ensuring integration across these models is a non-trivial task. Without strong automation support, scaling up ICME is going to be a difficult problem.

With this motivation, we are developing an IT platform called PREMΛP [4, 5] at Tata Consultancy Services. Our goal is to use this platform to industrialize the benefits of the ICME approach, with a special focus on integrated design of products and materials. In view of the vast diversity of material systems and component/product application categories, the platform consists of a set of domain dependent and domain-independent components as shown in Figure 1.

On the right side of the figure are the components that are domain dependent and those on the left are domain independent. A domain may refer to a material category with associated manufacturing processes and/or a product category. Domain specific components include models of various kinds, design templates, design rules, design cases, etc. Domain independent infrastructure includes, among other things, (a) knowledge engineering framework for

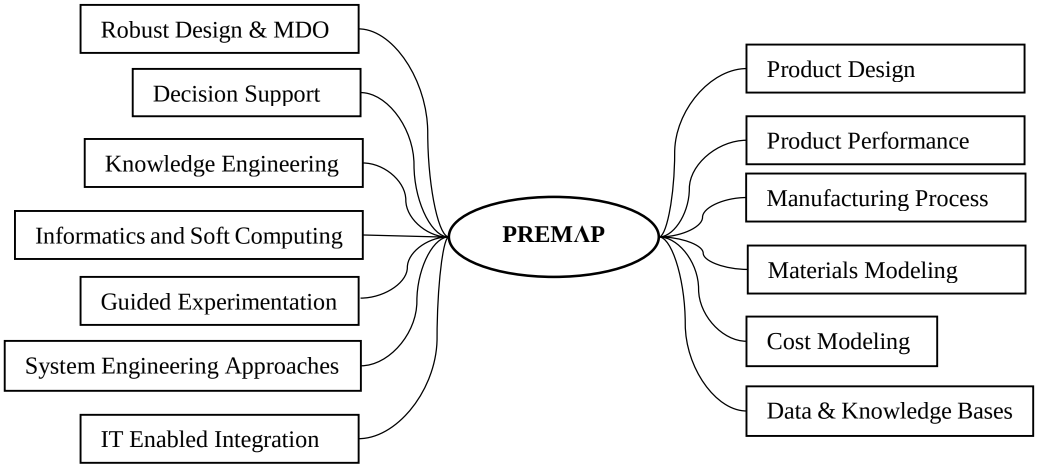

Figure 1. Domain independent (left) and domain dependent (right) components of the platform

knowledge management, (b) simulation services framework for simulation execution and simulation tool integration, (c) tools for robust design and multidisciplinary optimization techniques (MDO), (c) decision support tools (e.g., the compromise decision support problem construct), and (d) design of experiments and combinatorial experimentation tools to drive both simulation and experimental studies.

Building all these capabilities into the platform in an integrated manner requires a unifying semantic foundation. Domain ontology provides such a foundation. It serves as the common substrate for integrating different models. It serves as a means for capturing and organizing knowledge. However, ontology varies from subject to subject, and, being a generic platform, PREMΛP has to cater to a wide range of subjects. For instance, ontology of steel is different from ontology of a composite material. This calls for a flexible ontology engineering framework that enables us to create and evolve subject specific ontologies without hard coding them into the platform. We use model driven techniques to engineer such a framework. In this paper we present the modeling framework underlying the PREMΛP architecture and give a brief overview of some of the aspects automated using model driven techniques.

2. PREMΛP Modeling Framework

PREMΛP uses a reflexive modeling framework to bootstrap its modeling infrastructure.

2.1 Reflexive Modeling Framework

An information system can be seen as a collection of parts and their relationships. A model of an information system is a description of these parts and relationships in a language such as UML [9]. The modeling language itself can be described as a model in another language. The latter language is the meta-model for the former as shown in Figure 2.

We use a reflexive modeling language [7] that is compatible with OMG MOF [8] to define models at all levels. A model at each level is an instance of the model at the previous level. The model at level 1, the meta meta-model, is an instance of itself. The meta meta-model shown in Figure 2 is the base model. It is the schema for describing meta-models. The meta meta-model is capable of describing itself, i.e., it can model itself.

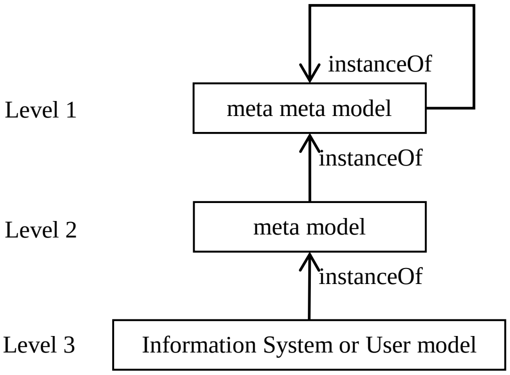

Figure 2. Modeling Layers

Every thing in a model is an object. An object is described by its class. A class is specified in terms of a set of attributes and associations. An object is an instance of a class that has attribute values and links to other objects as specified by its class. Since everything is an object, a class is also an object. A class is specified by another class called metaclass. In Figure 3, the class '*class'* is a metaclass which is an instance of itself. Any class that inherits from the class '*class'* is also a metaclass. A meta model specification consists of a model schema, which is an instance of the meta meta-model, and a set of constraints and rules to specify consistency and completeness checks on its instance models. Due to the reflexive nature of the meta-meta-model, there is no inherent limit on the number of modeling layers that can be supported. We use OCL [10] to specify well-formed-ness constraints over models. Cardinality and optionality constraints are supported by the reflexive model itself. We use an industrial-strength relational database as a storage mechanism for managing large scale models. Storage schema reflects the structure of models.

2.2 Ontology Modeling Framework

In PREMΛP, ontologies can be classified into a set of subject areas, such as materials, products, manufacturing processes, etc. Each subject area contains ontologies of subjects that belong to that area.

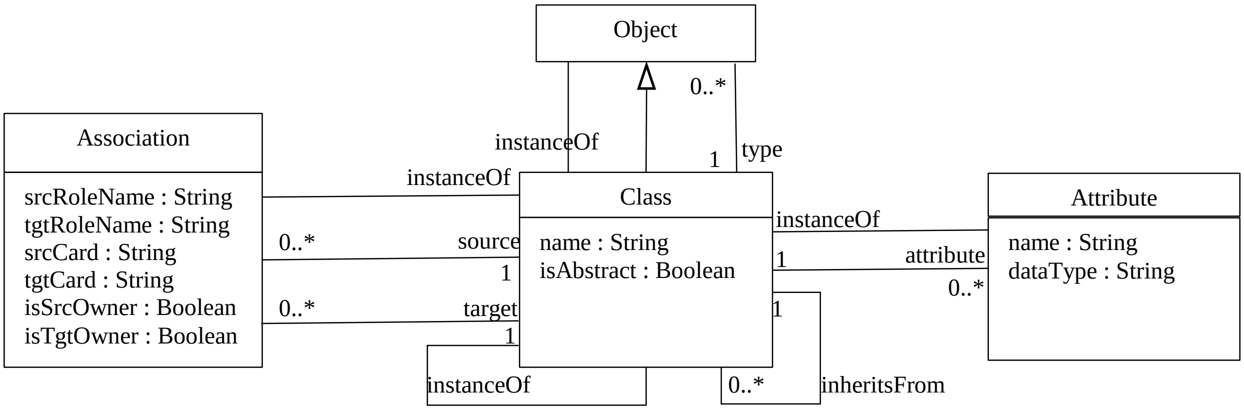

Figure 3. Reflexive Meta Meta-Model

For instance, materials subject area contains ontologies of steel, composite materials, etc. In the context of PREMΛP, while we know the subject areas we want to support, upfront we do not know all the specific subjects that we want to support. That depends on the problems we want to solve on the platform, which is open ended. So we cannot hard-code subject specific ontologies into the platform. Instead they should be treated as first-class entities – i.e. it should be possible to create, modify and delete them on a need basis. To address this, we have conceptualized domain models at two ontological levels - a meta level and a subject level, as shown in Figure 4.

As mentioned above, models in ICME can be broadly categorized into three subject areas - materials, products and processes. Corresponding to these subject areas we have three related meta models -- materials meta model, products meta model and process meta model. Essentially, a meta model can be viewed as defining a language for a subject area, using which subjects in that area can be described. For instance, materials meta model provides the language to describe materials. Subject specific ontologies are created as instances of these meta models. For instance, steel ontology is created as an instance of the materials meta model, gear ontology is created as an instance of the products meta model, and so on.

We illustrate this with an example. Figure 5 shows a part of the component meta model, which is a part of the products meta model. A component has a geometry and a set of functional and geometric features. These features may be described in terms of a set of parameters. A component may be made from one or more materials; similarly different geometric features of the component may be made from different materials.

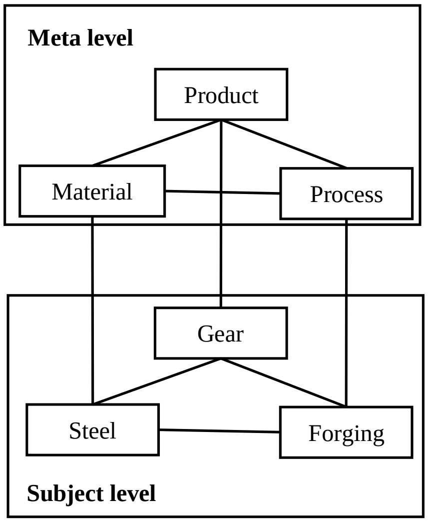

Figure 4. Domain Ontology Levels

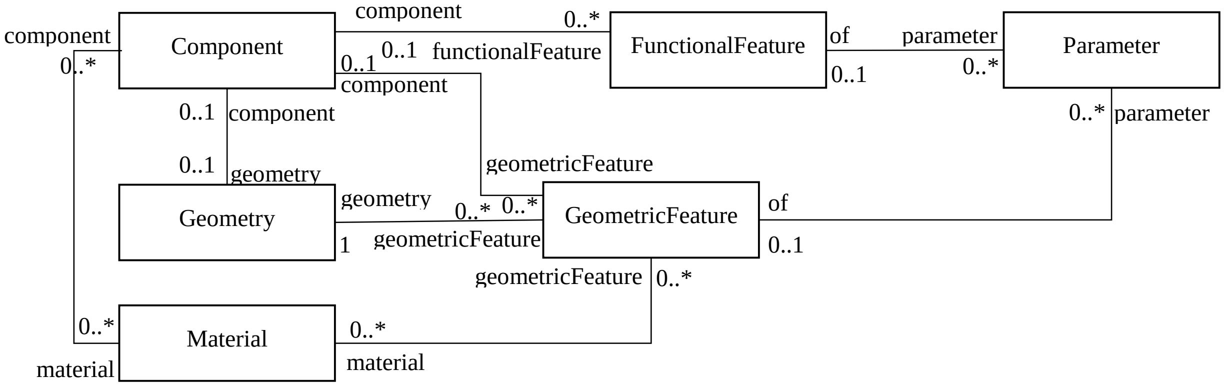

Figure 5. Component Meta Model

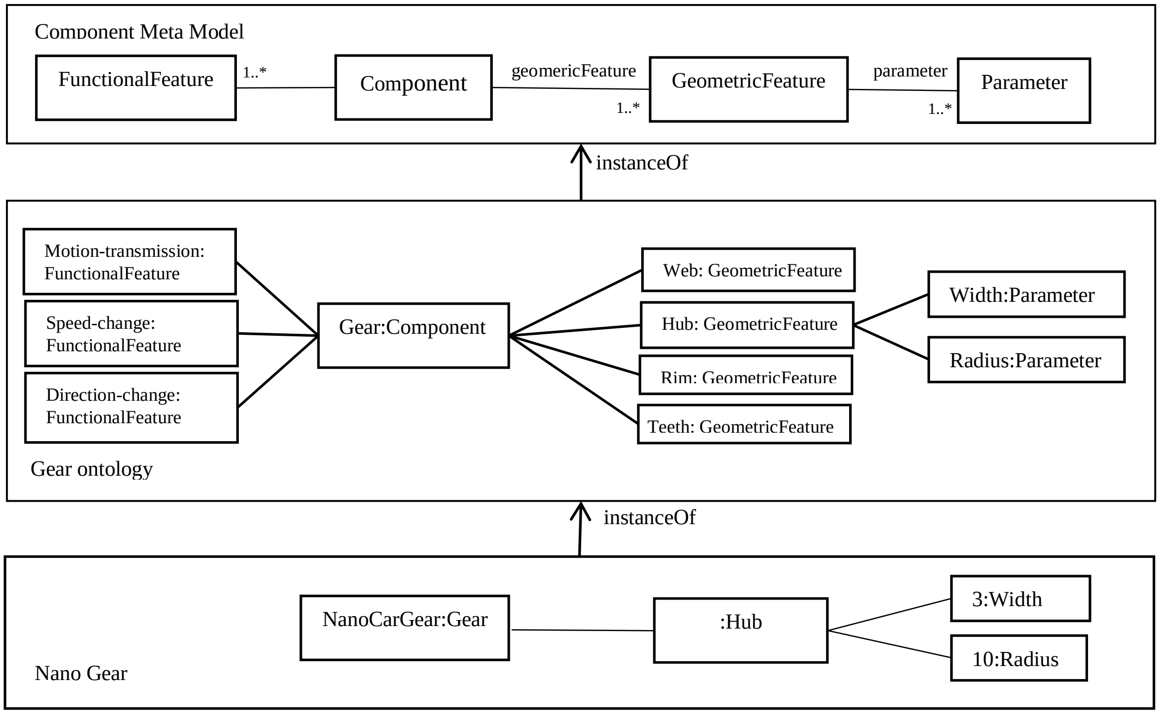

Figure 6. Component Modeling Layers

Figure 6 shows Gear ontology as an instance of this meta model. A gear is a component whose geometry has features such as hub, web, rim and teeth. Its function is to transmit motion in the same or a different direction and a change in rotational speed. The geometric feature 'hub' has diameter and width as parameters (parameters of other features are omitted from the diagram). The figure also shows a specific gear (NanoCarGear) with its dimensions, as an instance of the gear ontology.

This layered modeling architecture provides two benefits:

1) It provides a means to organize domain knowledge systematically. Knowledge that is applicable across all subjects of a subject area is captured at the meta model level; knowledge that is specific to a design subject is captured at the subject model level; and knowledge that is very specific to a design instance is captured at the instance model level. To give a trivial example, with reference to the meta model in Figure 4, we have a constraint that says that the materials used for a geometric feature of a component must be a subset of the materials allowed for the component. This applies to all types of components. Similarly, taking an example at the subject model level, we may have a rule that specifies what type of forging process to use for a gear. This applies to all gear design instances. Thus we could capture knowledge at different levels across different subject areas. This knowledge can be used not only to guide a designer in making right decisions, but also to ensure integration across design domains.

2) It lends extensibility to the platform, by enabling new subjects to be created as instances of meta models. For instance, to extend the platform to support the design of composite materials, we create composites ontology as an instance of the materials meta model. Similarly to support the design of an engine block, we create engine block ontology as an instance of the products meta model. Subject specific ontologies thus become first class entities in the platform.

3. Model Driven Engineering in PREMΛP – A Few Examples

Model driven engineering is used extensively to automate various aspects within the platform. We give a brief overview of a few of these.

3.1 Simulation Tool Integration

A design workflow consists of design of several process steps such as forging, machining, carburization, quenching, tempering, etc. Each of these processes has its own simulation model. In integrated design simulation, these models have to be simulated in an integrated manner, with right information flowing from one model to the other [3, 6]. This is done by mapping the inputs and outputs of each simulation tool to the domain ontology, as shown in Figure 7.

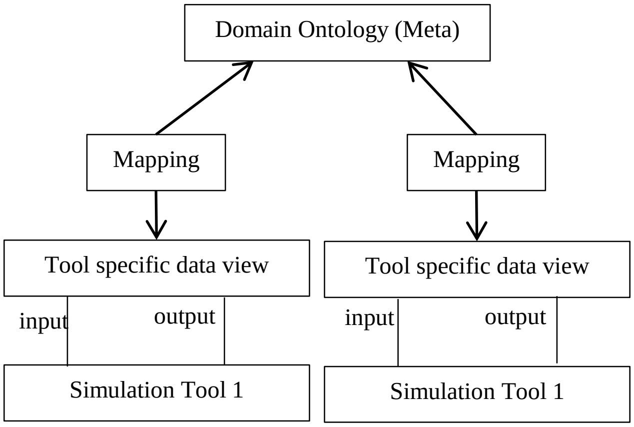

Figure 7. Simulation Tool Integration

It is then possible to validate a process chain for information integrity by checking that right information is flowing to the right process step. From these mappings it is also possible to generate input/output adapters for plug-and-play integration of simulation tools. These mappings are specified at the meta model level. As a result, once a tool is integrated into the platform, there is no need to write separate adapters for each subject separately. For instance, once a finite element simulation tool is integrated at the meta level, we don't have to write separate adaptors for gear simulation, clutch simulation, etc.

3.2 Data Layer Automation and Virtualization

Data of different subject areas might be stored in different physical stores. Depending on volumes, data characteristics and performance requirements, different storage mechanisms, such as relational database, object databases, graph databases etc., might be better suited for different subject areas. The architecture should be flexible enough to support different storage mechanisms and to change them on a need basis. We use model driven generation to achieve this flexibility. The architecture should also provide a uniform data access interface. As shown in Figure 8, we map our domain ontology to data models of physical storage structures. These mappings are specified at the meta model level. From these mappings we generate a data access layer. Interfaces remain uniform as they are defined in terms of the domain ontology; only the implementations change according to the storage technologies.

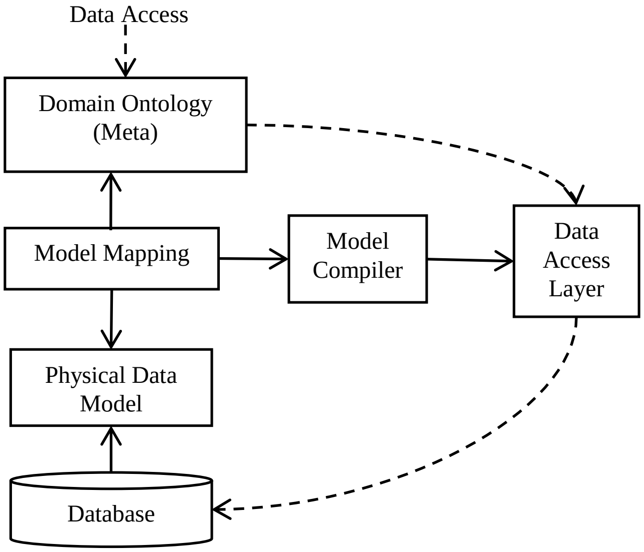

Figure 8. Data Virtualization

3.3 User Interface Generation

A design workflow may contain multiple screens for user interaction. We generate these screens using model driven techniques. These screens are defined for specific subject models and get their data from corresponding instance models. The data view of the screens are encoded using screen specific view models. There is a two-way synchronization between the screen elements and the view model. Whenever view model changes, the screen is updated and whenever user specifies some values in screen controls, the view model is updated. The interaction between the screens and the database is performed through PREMΛP platform services. The view model elements are mapped to the service messages, which are defined using subject ontology elements. The input messages for the services are constructed from the mapped view models. The view models are updated in response to the output messages from the services. The relations between PREMΛP services, subject model, view model and the screen elements are shown in Figure 9. GUI screen implementations are generated from these models.

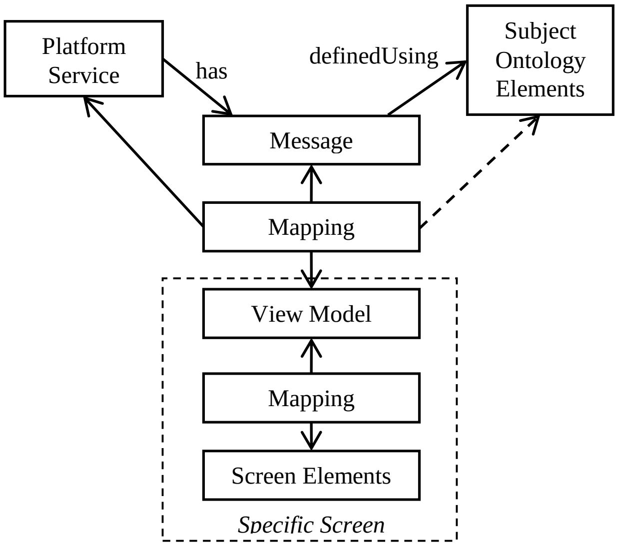

Figure 9. User Interface Generation

3.4 Data Integration

Data integration techniques are used in PREMΛP to utilize available information about the materials or processes. The data sources may include laboratory databases, factory floor databases, or third party proprietary data. These data sources are individually mapped to subject model ontology using Global-as-view (GAV) [15, 16] or Local-as-view (LAV) [17] schemes. The subject model is treated as the unified conceptual model describing all the data sources. A query on the subject model gets converted to a DFG (data flow graph). The DFG is responsible for extracting data from individual sources and suitably combining them to produce the query result.

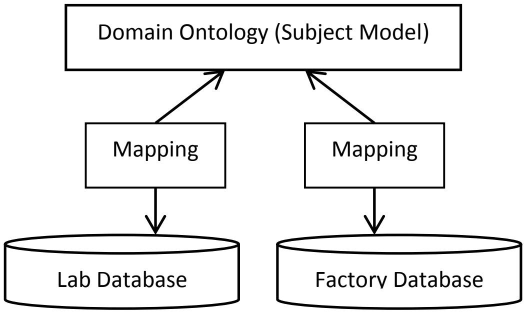

Figure 10. Data Integration

4. Related Work

Model driven engineering is growing in popularity. Several large enterprise scale applications have been developed using MDE techniques [7]. Object management group (OMG) has developed a number of standards in this space under its model-driven architecture (MDA) [14] initiative. While OMG promotes UML [9] as the de-facto modeling standard, experience shows that a multi-modeling approach, where different purpose specific models are used for different aspects, scales up much better in practice [7].

Especially when engineering a platform such as PREMΛP, where a large number of diverse sets of concepts and mechanisms have to be integrated, one needs a multi-layered modeling approach such as the one discussed in this paper.

Ontology modeling approaches such as OWL [11] are also growing in popularity. OWL has three sublanguages: OWL Lite, OWL DL and OWL-FULL. Of these, OWL Lite and OWL-DL only support models at two levels. This is insufficient for an extensible platform such as PREMΛP where subject specific ontologies are first class entities. OWL-FULL allows a class to be an instance of another class. However, there are no OWL Full reasoners available [12, 13]. Besides, in a platform engineering scenario, models should not only capture domain semantics, but also various engineering aspects of the platform. What we need is a combination of the flexibility of model driven engineering principles and the deductive reasoning capabilities of ontologies.

5. Summary

We have given an overview of a computational platform that we are developing in the engineering design space and briefly discussed the model-driven engineering design principles underlying its architecture. We have identified the domain modeling challenge and presented a modeling framework that has been developed to address this challenge. We have also given a brief overview of how model driven techniques have been used to automate some of the key features. There are many other features such as the knowledge engineering framework which have not been discussed due to space limitation.

6. REFERENCES

[1] NRC Report (2008) Integrated Computational Materials Engineering: A Transformational Discipline for Improved Competitiveness and National Security. The National Academies Press, National Research Council, Washington, D.C.

[2] TMS Study Report on Integrated Computational Materials Engineering (ICME) – Implementing ICME in the Aerospace, Automotive and Maritime Engineering, (2015) TMS, http://www.tms.org/ICMEStudy.

[3] KONTER, A.W.A., FARIVAR, H., POST, J. and PRAHL, U. Industrial Needs for ICME. JOM: the journal of the Minerals, Metals & Materials Society, 2015.

[4] Bhat, M., Shah, S., Das, P., Kumar, P., Kulkarni, N., Ghaisas, S. S. and Reddy, S. S. (2013), PREMΛP: Knowledge Driven Design of Materials and Engineering Process, A. Chakrabarti and R. V. Prakash (eds.), ICoRD’13, Lecture Notes in Mechanical Engineering, Springer India, pp. 1315-1329.

[5] Gautham, B.P., Singh, A.K., Ghaisas, S.S., Reddy, S. S. and Mistree, F. (2013a) PREMΛP: A Platform for the Realization of Engineered Materials and Products, A. Chakrabarti and R. V. Prakash (eds.), ICoRD’13, Lecture Notes in Mechanical Engineering, Springer India, pp. 1301-1313.

[6] Tennyson, G., Shukla, R., Mangal, S., Sachi, S. and Singh, A.K. (2015), ICME for process scale-up: Importance of vertical and horizontal integration of models, Proceedings of the 3rd World Congress on Integrated Computational Materials Engineering ICME’15, 11-21.

[7] Vinay Kulkarni, Sreedhar Reddy, Asha Rajbhoj: Scaling Up Model Driven Engineering - Experience and Lessons Learnt. MoDELS (2) 2010: 331-345

[8] Model Object Facility, http://www.omg.org/spec/MOF/2.0

[9] Unified Modeling Language, http://www.omg.org/spec/UML/2.2/

[10] Object Constraint Language, http://www.omg.org/spec/OCL/2.2

[11] OWL, Web Ontology Language, http://www.w3.org/TR/owl-guide/

[12] http://www.w3.org/2001/sw/wiki/OWL/Implementations

[13] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

[14] Model Driven Architecture, http://www.omg.org/mda/

[15] Ullman, J. D. Information integration using logical views. Database Theory—ICDT'97. Springer Berlin Heidelberg, 1997. 19-40.

[16] Lenzerini, Maurizio. Data integration: A theoretical perspective. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002.

[17] Halevy, Alon Y. Answering queries using views: A survey. The VLDB Journal10.4 (2001): 270-294.