AN INTEGRATED FRAMEWORK FOR MODEL-DRIVEN PRODUCT DESIGN AND DEVELOPMENT USING MODELICA

Adrian Pop[1] **Olof Johansson**[2] **Peter Fritzson**[3]
Programming Environments Laboratory
Department of Computer and Information Science
Linköping University
58131 Linköping
Sweden

Abstract

This paper presents recent work in the area of model-driven product development processes. The focus is on the integration of product design tools with modeling and simulation tools. The goal is to provide automatic generation of models from product specifications using a highly integrated set of tools. Also, we provide the designer with the possibility of selecting the best design choice, verified through (automatic) simulation of different implementation alternatives of the same product model. To have a flexible interaction among various tools of the framework an XML representation of the Modelica modeling language called ModelicaXML is used. For efficient search in a large base of simulation models the Modelica Database was designed.

1 Introduction and Related Work

Designing products is a complex process. Highly integrated tools are essential to help a designer to work efficiently. Designing a product includes early design phase product concept modeling and evaluation, physical modeling and simulation and finally the physical product realization. For conceptual modeling and physical modeling and simulation available tools provide advanced functionality. However, the integration of such tools is a resource consuming process that today requires large amounts of manual, and error prone work. Also, the number of physical models available to the designer in the product concept design phase is typically quite large. This has an impact on the selection of the best set of component choices for detailed product concept simulation.

To address these issues we have integrated new product concept design tools with physical modeling and simulation tools in a framework for product design. In our proposed framework, the product concept design phase of the product development process is based on Function-Means tree decomposition [7, 13]. This phase is implemented in a first version of a prototype tool called FMDesign, developed in cooperation with the Machine Design Group led by Petter Krus, IKP, Linköping University.

As an example of Function-Means tree decomposition we give a landing function in an airplane. This function can be represented by two different means: hydraulic landing gear or electric landing gear. Each of the two alternatives can be selected and configured to simulate its properties.

Starting from FMDesign tool, our integration work extends the framework in two ways:

- Providing a S*election and Configuration Tool* that helps the designer to choose a specific implementation for the means in the function-means tree from a Modelica model/

[1] Phone: +46 13 285781, Fax: +46 13 284499, Email: adrpo@ida.liu.se
[2] Fax: +46 13 284499, Email: olojo@ida.liu.se
[3] Phone: +46 13 281484, Fax: +46 13 284499, Email: petfr@ida.liu.se

component database. This tool also provides component configuration and has links to a Modelica standard based simulation environment for component editing.

- Providing an A*utomatic Model Generation Tool* that helps the designer to choose the best implementation from different design choices by evaluation through simulation of automatically generated models of candidate product concepts. If the designer is not pleased with the results, he/she can either implement new models for the components that did not perform in the desired way or reiterate in the design process and choose other alternatives for implementing different functions in the product, or change the configuration parameters for models at deeper levels of detail.

The paper is structured as follows: The next section presents an overview of our proposed framework. Section 3 enters in the details of the framework components and their interaction. Section 4 presents our conclusion and future work.

The presented system has similarities with the Schemebuilder tool [8]. However our work is more oriented towards the design of advanced complex products that require systems engineering, and targeted to the simulation modeling language Modelica, which to our knowledge has more expressive power in the areas of our research, than many tools for systems engineering that are currently widely used. For details on Systems Engineering, see [2].

2 Architecture overview

The architecture of our extended framework is presented in Figure 1. The entire product concept design process is iterative.

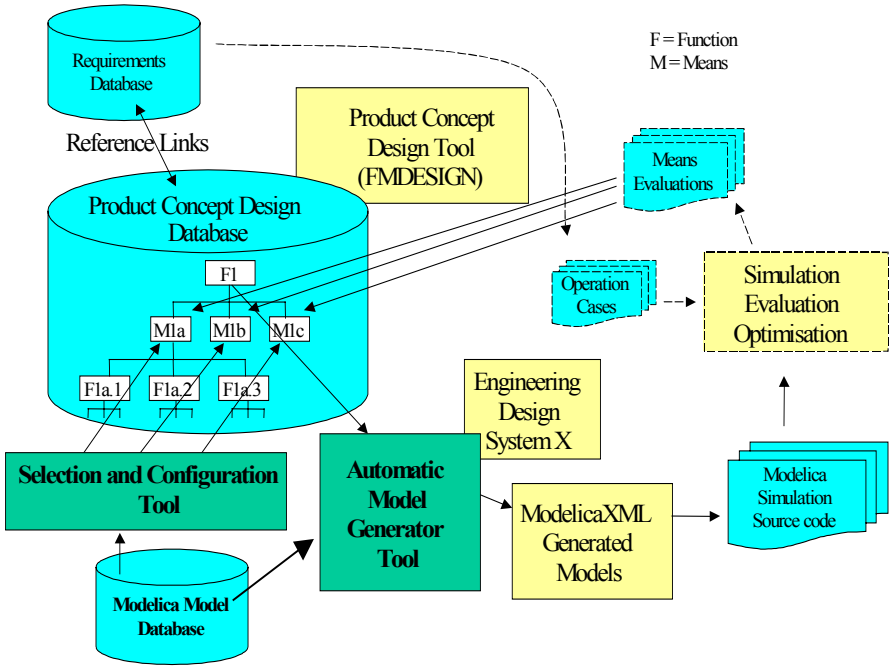


Figure 1: Design framework for product development

Starting from requirements for a product the designer will use the FMDesign prototype for modeling alternative product concepts. The knowledge base for designing a product is organized into function-means trees. A function in the product can be realized by alternative means. A product concept is a set of means that document selected solution alternatives for implementing the functions in a product concept. Example of a function is `"Actuator Power Supply"`, with means `"Hydraulic Power Supply"` or `"Electrical Power Supply"`. Means must be implemented by (physical) components arranged in a bill-of-material like tree of implementation objects.

One can roughly say that a means and its implementation are the same, but at different levels of detail. Implementation objects (not

shown in the figure) may represent existing component products on the market or manufactured components. Implementation objects carry data that is important for the product concept design, and references to more detailed design information like CAD-drawings, simulation models etc. Some (physical components) may implement several means, like an aircraft wing that creates lift and stores fuel.

To map suitable simulation model implementations to a means, the designer would use the Modelica Database query facility provided by the *Selection and Configuration Tool*. This tool also provides configuration of the simulation components and uses the desired Modelica environment for component editing.

When the product concept design phase of the product is sufficiently complete, the designer can generate code for simulation from the implementation tree using the *Automatic Model Generator Tool*. The generator will output models (different versions for different product concepts) in ModelicaXML. From Modelica-XML the models are translated to Modelica to be simulated. The designer can review the simulation results in tools like MathModelica [3], Dymola [1] or OpenModelica [10] and then selects (in FMDesign) the desired model alternative for the implementation. If the designer sees that some means do not perform in the desired way, a customized simulation model can be built, or a search conducted for more alternatives for that specific means.

3 Detailed framework description

In this section we present the tools from our proposed framework. Also, we briefly explain in each section how they interact.

3.1 ModelicaXML

Modelica [4, 9] is an object-oriented language used for modeling of large and heterogeneous physical systems. For modeling with Modelica, commercial software products such as MathModelica [3] or Dymola [1] have been developed. However, there are also open-source projects like the OpenModelica Project [10].

Modelica is translated to ModelicaXML using a Modelica parser (Figure 2).

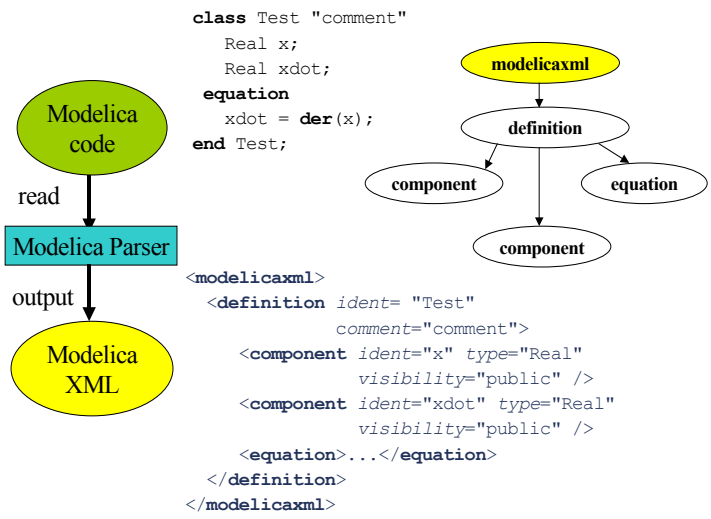


Figure 2: Modelica and the corresponding ModelicaXML representation

ModelicaXML represents an XML serialization of the Abstract Syntax Tree of the Modelica language obtained after the parsing. In our framework, ModelicaXML is used as an interchange format between the different design tools.

The advantages of having an alternative representation for Modelica in XML are:

- Flexible interaction and translation between different types of physical modeling languages and modeling tools. Also, easy generation of model documentation.
- Basic search and query functionalities over models.
- Easy transformation and composition of models [12].

For more information on ModelicaXML the reader is referred to [11] and [9].

3.2 Modelica Database (ModelicaDB)

The features of the Modelica language and Modelica tools has made easy for designers to create models. Also, the Modelica community has a growing code-base. In order to cope with interoperability between Modelica and other modeling languages we first developed ModelicaXML. However, scalability and efficient search features for XML require extensive skills in vendor specific products. To quickly get such features without taking on that huge learning

effort, we have designed the Modelica Database (ModelicaDB).

The Modelica Database is populated with Modelica models and libraries by importing their ModelicaXML representation. The UML model of this database is presented in the Appendix. For paper space reasons we use a somewhat customized compressed graphical representation of UML class diagrams, where inheritance is represented with a box between the class name and attributes box, where inherited super classes are preceded with a "->". For details on UML see [6].

Here we briefly explain the most important structures. They are tightly coupled with the Modelica structure [9, 11]:

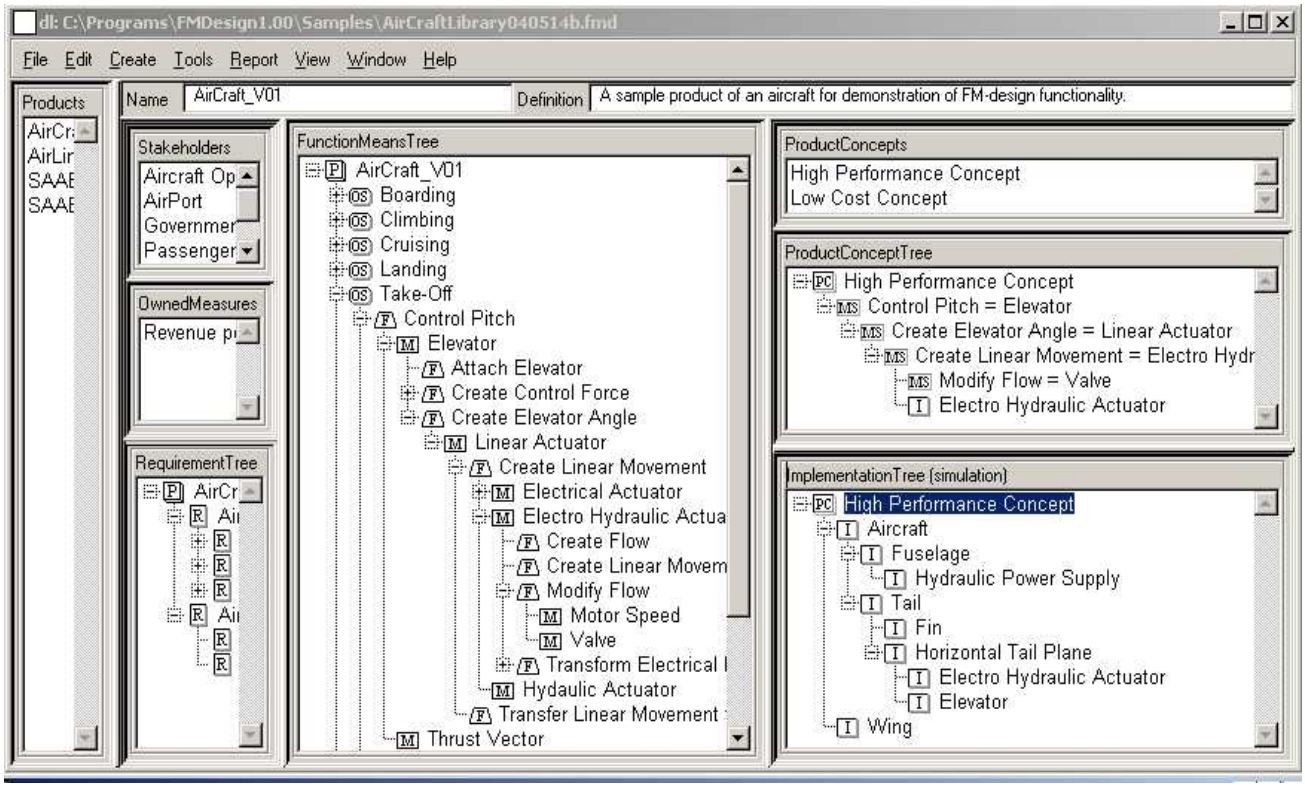
- *Modelica Repository*: contains several Modelica Models.
- *Class*: A class represents the fundamental model element from the Modelica language. It can include several *Component* clauses, *Equation* and *Algorithm* statements. The component sections can be declared as public or private in order to provide only the desired interface to the outer world. Specifying that the equation or algorithm sections are only active at the initialization phase they can be declared as initial.
- *Component*: used to define parameters, variables, constants, etc to be used inside a class.
- *Equations and Algorithms* are used to specify the desired behavior for a class.

In the product design framework the role of ModelicaDB is to provide searching and organization features of a large base of simulation models. This base grows with every product model developed or with the import of additional simulation models from other sources (i.e. the Modelica community). For example, if we want to obtain all the models that have certain parameter names we have to search in the database for all classes that have a component with the attribute `variabilityPrefix` set to `"parameter"` and has the specified name. These searches will be integrated in FMDesign using dialogs and completely transparent for the user.

3.3 FMDesign

The FMDesign (Figure 3) prototype tool helps the designer in creating product specifications using function-means trees.

Figure 3: FMDesign



The created product model is stored in a product design library for later reuse. Throughout the product concept design process the designer can use the existing concepts stored in the product design library in order to model the desired product. A somewhat simplified meta-model of the information structure edited in FMDesign is presented as an UML class diagram in the appendix section.

In the framework, FMDesign is the central front-end to specific components. FMDesign delegates searches in the ModelicaDB using the *Selection and Configuration Tool* and it uses the *Automatic Model Generation Tool* to generate the models for simulation.

As we can see in Figure 3, the work area is divided into several parts:

- *Products*: Here products are created, deleted and selected. When a product is selected, the trees owned by it and described below, are displayed.
- *Requirements Tree*: in this view the requirements for a product can be specified.
- *Function-Means Tree*: in this view the designer can define the operation states, functions, their alternative means etc, of the selected product.
- *Product Concepts*: Allows creating, deleting and selecting product concepts.
- *Product Concept Tree*: displays the currently selected Product Concept Tree, and allows the user to select which means that will implement different functions in the product, using drag-drop. Selected means can be customized for the current product concept by overriding the default values for its design variables owned by a selected means.
- *Implementation Tree*: displays and provides functionality for editing one of many configurable Implementation Trees for the currently selected product concept. These implementation trees organize the implementation objects that represent and refer to more detailed models of physical objects, functional models, simulation models, geometrical layout models etc, and organize them into trees that are useful for interfacing with tools later in the product development process.

We only use the Implementation Tree of type simulation to generate the Modelica simulation model for a product. The Implementation Tree of type geometrical can be used in the visualization of the product.

3.4 The Selection and Configuration Tool

The S*election and Configuration Tool* extends the framework by adding integrated search capabilities in FMDesign. The tool is coupled with the Implementation Tree for a Product Concept. The designer uses the selection tool to search (query) the Modelica Database for desirable simulation components to implement a certain means. An implementation object in the simulation implementation tree represents the selected simulation component. Simulation component to means mapping reflects the various design choices made by the designer. In this way, the designer can experiment with different simulation component implementations at various level of detail for a specific means. When choosing alternatives for a specific means the designer has two possibilities: to browse the repository of simulation models classified according to physical concepts or to use the search dialog. The search dialog provides the following functionality:

- Textual/pattern search of components, search for a component in a specific physical domain, search for a component with specific parameters.
- Adding/deleting a product concept specific means to simulation component mapping where the simulation component is referred from an implementation object.

After building the means-component mappings the designer can choose to edit or configure components by using the configuration dialog that provides the following functionality:

- Set implementation component parameters or parameters ranges.
- Edit the simulation component in the desired Modelica environment and use the edited component, which is also automatically added to the Modelica Database.

3.5 The Automatic Model Generator Tool

The Automatic Model Generator Tool provides the second extension of the framework.

The model generator tool has as input the Implementation Tree (Figure 3, lower right) of a product and as output the complete simulation model with the alternative design choices.

The automatic model generator traverses the Implementation Tree of a Product Concept and outputs ModelicaXML models by choosing the combination of selected components for means. The generated models are then translated to Modelica for means evaluation through simulation. To simulate the models, commercial tools like Dymola and MathModelica or the open-source OpenModelica [10] compiler can be used.

After the simulation of the generated models, the results are used as feedback for the designer. Using this feedback the designer can then choose the best-suited model, based on the simulation results.

4 Conclusions and Future Work

As future work we want to explore the use of ontologies for product concept design and for the classification of the available component libraries.

The languages developed by the Semantic Web [5] community will be used. Research efforts based on this standard are integrating experience of many promising research areas, for instance declarative rules, which still lack a vendor neutral exchange formats for industrial applications. The semantic web standard lacks important functionality for quality assurance and other necessary functionality, which today is implemented in commercial products, but will open up for sharing of important research results with industry in collaborative environments. Also we would like to improve the *Automatic Model Generator Tool* by using parts of the composition and transformation framework described in [12].

In the future we want to provide automatic evaluation through simulation of the generated models based on the constraints collected from the Product's Requirement Tree.

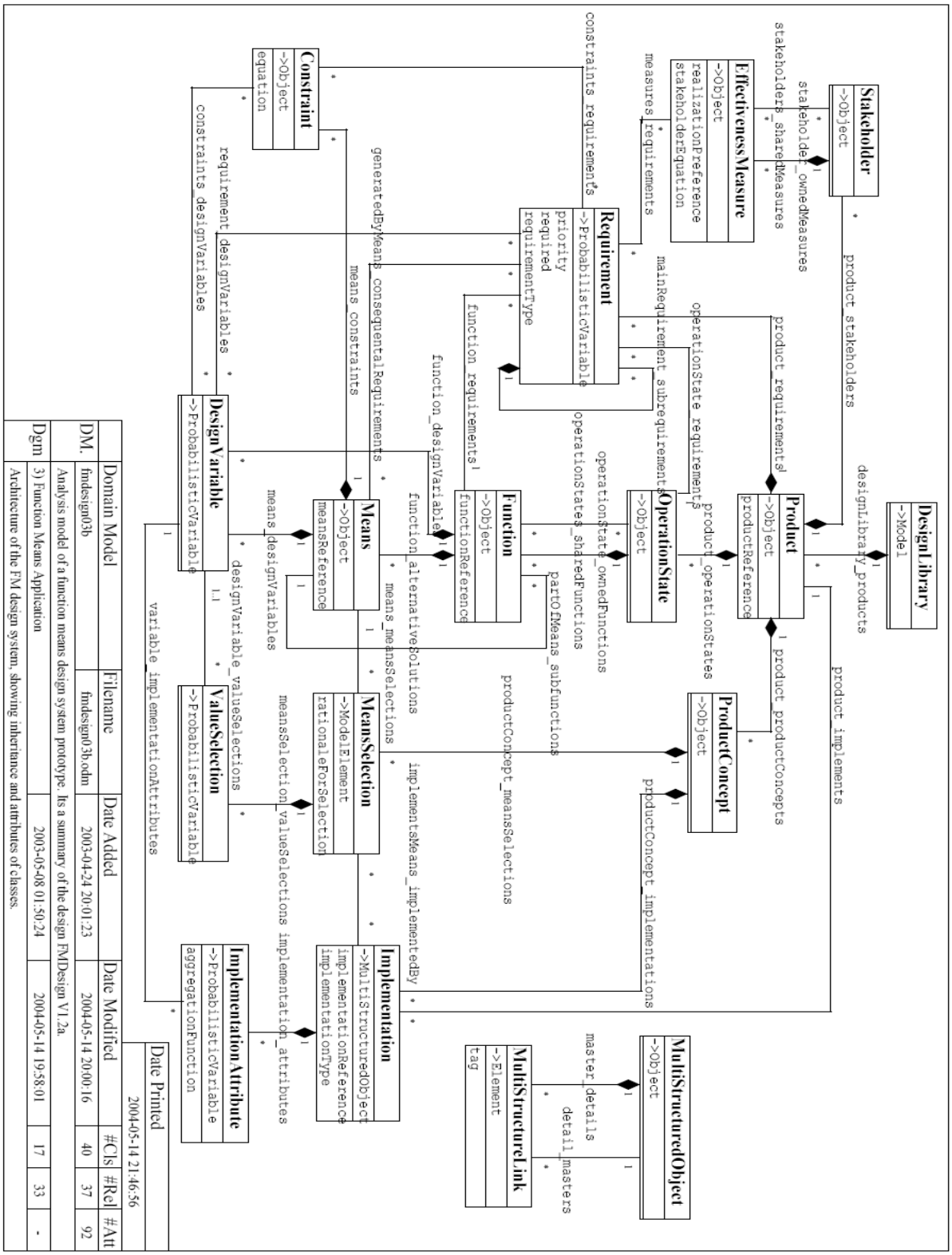
5 Acknowledgements

The ProViking research program, created by the Swedish Foundation for Strategic Research supported this research through the project *System Engineering and Computational Design (SECD)*. *The National Computer Graduate School in Computer Science (CUGS)* and *Vinnova* through the *Semantic web for products (SWEBPROD)* project.

6 References

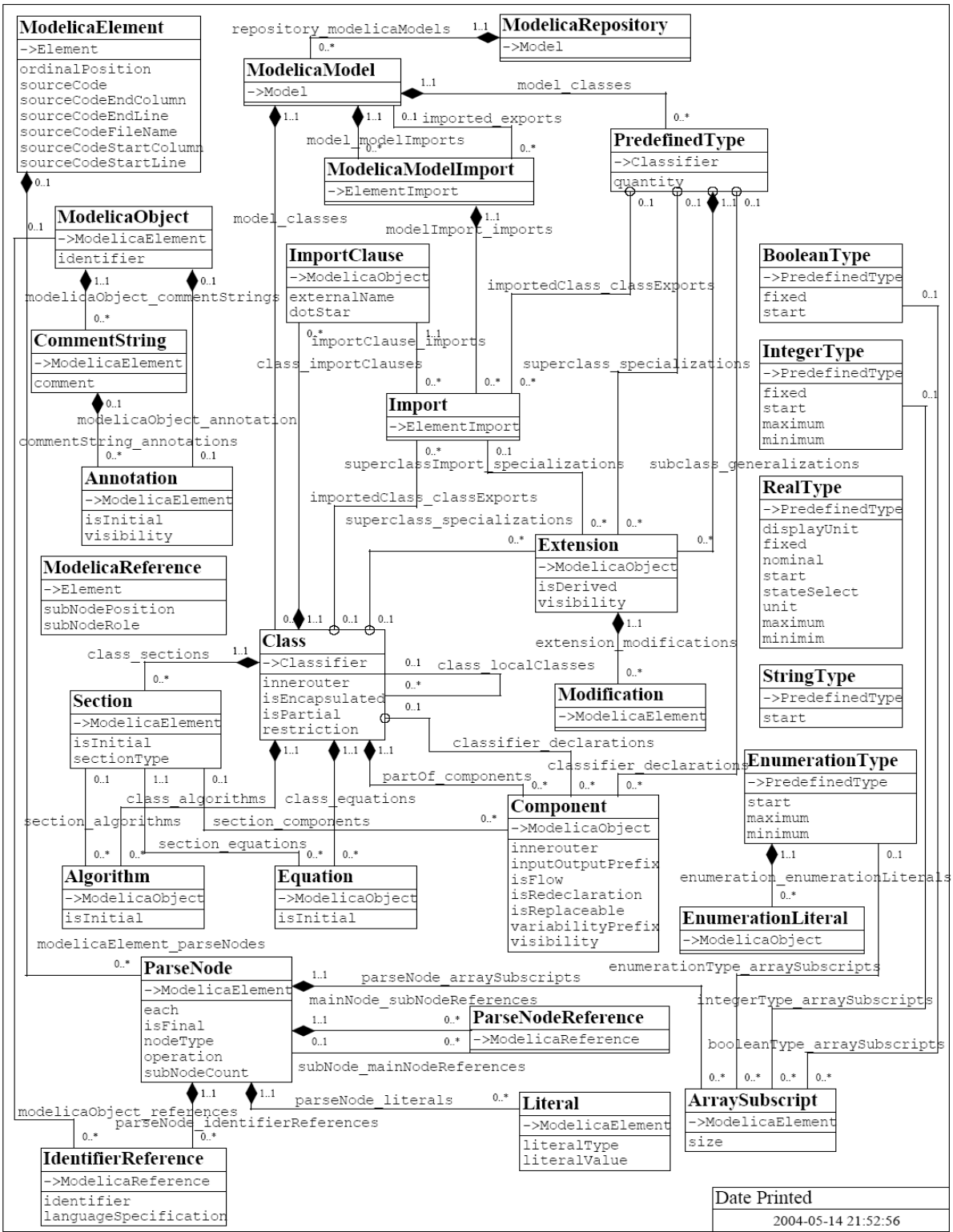
1. Dynasim. *Dymola*, http://www.dynasim.se/.
2. INCOSE. *International Council on System Engineering*, http://www.incose.org.
3. MathCore. *MathModelica*,http://www.mathcore.se/.
4. *Modelica: A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification version 2.1*, Modelica Association, 2003,
5. *Semantic Web Community Portal*, http://www.semanticweb.org/.
6. OMG. *Unified Modeling Language*, http://www.omg.org/uml.
7. Mogens Myrup Andreasen. *Machine Design Methods Based on a Systematic Approach (Syntesemetoder på systemgrundlag)*, Lund Technical University, Lund, Sweden, 1980.
8. R.H. Bracewell, D.A.Bradley. *Schemebuilder, A Design Aid for Conceptual Stages of Product Design*, in *International Conference on Engineering Design, IECD'93* 1993, The Hague.
9. Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, Wiley-IEEE Press, 2003, http://www.mathcore.com/drmodelica.
10. Peter Fritzson, Peter Aronsson, Peter Bunus, Vadim Engelson, Levon Saldamli, Henrik Johansson and Andreas Karström. *The Open Source Modelica Project*, in *Proceedings of The 2th International Modelica Conference*,March 18-19, 2002, Munich, Germany.
11. Adrian Pop, Peter Fritzson. *ModelicaXML: A Modelica XML representation with Applications*, in *International Modelica Conference*,3-4 November, 2003, Linköping, Sweden.
12. Adrian Pop, Ilie Savga, Uwe Assmann and Peter Fritzson. *Composition of XML dialects: A ModelicaXML case study*, in *Software Composition Workshop 2004, affiliated with ETAPS 2004*,3 April, 2004, Barcelona.
13. Sören Wilhelms, *Reuse of Principle Solution Elements In Conceptual Design.* Lic. thesis 1044, Linköping University, 2003

7 Appendix



	Domain Model	Filename	Date Added	Date Modified	#Cls	#Rel	#Att
DM.	fmdesign03b	fmdesign03b.odm	2003-04-24 20:01:23	2004-05-14 20:00:16	40	37	92
	Analysis model of a function means design system prototype. Its a summary of the design FMDesign V1.2a.						
Dgm	3) Function Means Application		2003-05-08 01:50:24	2004-05-14 19:58:01	17	33	-
	Architecture of the FM design system, showing inheritance and attributes of classes.						

Date Printed
2004-05-14 21:46:56



Date Printed
2004-05-14 21:52:56

	Domain Model	Filename	Date Added	Date Modified	#Cls	#Rel	#Att
DM.	ModelicaDB-21-02n	ModelicaDB-21-02o.odm	2003-03-04 08:03:07	2004-05-14 21:51:54	53	50	137
	Modelica 2.1 database design for ModelicaXML information exchange. The goal is a repository that can be round-tripped with Modelica source code.						
Dgm	SIMS'2004: Modelica Package		2003-03-04 09:33:10	2004-05-14 21:51:54	29	39	-
	Diagram 1: UML Model of Modelica 2.1. (Note that "->ModelicaObject", means inherits from ModelicaObject)						