CONFIGURATION MANAGEMENT OF MODELS FOR AIRCRAFT SIMULATION

Henric Andersson* **, Sören Steinkellner* **, Hans Erlandsson*
*** Saab Aerosystems, **Linköping University**

Keywords: *Software Product Line, Modularity, Configurator, Simulation*

Abstract

Support for configuration and instantiation of large-scale aircraft simulations has become a major issue as the numbers of models grow, model fidelity increases and there is a trend to design models to allow reuse between simulation environments. In this work a method for configuration support is presented that is based on the Product Line principles with structures and data inherited from the Product Data Management system. An XML-based information object to carry product configuration data and knowledge between tools, called a CNA-string is introduced. A rule-based method to support specification of consistent configurations is adopted from the sales configuration domain. The application example is configurations of the Gripen fighter aircraft simulation models.

1 Introduction

Increases in computer performance enable modeling and simulation based approaches to be used more extensively in the development of products and systems. Simulation is also used for training and to support certification activities. Large-scale rigs and simulators rely increasingly on software models instead of hardware components, mainly due to the cost per simulation hour. Management of models is identified as a growing concern with partly new needs to support engineers and decision makers.

This paper describes the challenges of model management for large-scale simulation in more detail to describe the fundamental needs in product line development supported by modeling and simulation. Some basic definitions are given as a foundation to develop the future simulation configuration methodology. The hypothesis presented in this paper is that Configuration Management (CM) of parametric models and other simulator components can be efficiently handled through the Software Product Line (SPL) approach, enhanced by Product Data Management (PDM) methodology/systems.

One distinction between software product line engineering and conventional software engineering is the presence of variation of the software artifacts. In the early stages of SPL engineering, software artifacts are designed to contain variations that represent choices about how a final software product will behave. At a point during the engineering process, decisions are made for the variations, after which the behavior of the final software product is fully specified [12]. This is illustrated in Fig. 1.

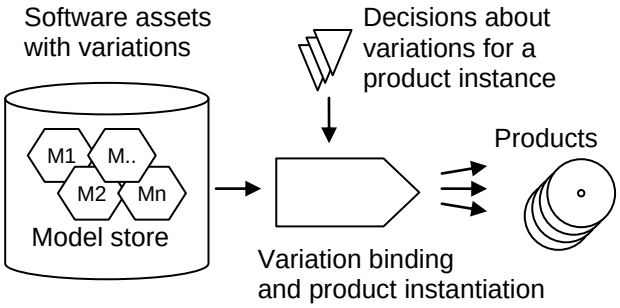

Fig. 1. Core concept of product instantiation in a Product Line approach

A specific property of the *simulation* software product line described in this work, is that a portion of it is a representation of another product line; viz the aircraft product family. The components of the simulation SPL do not in general have exactly the same functionality as the represented components. The models are

enhanced with e.g. fault simulation functions, but are simplified in other respects. This implies that variations and combinations of the simulation models (SPL artifacts) are partly constrained or guided by the variability rules of the aircraft's components and functions.

1.1 Challenges

The following challenges in set-up and support of large-scale simulations have been identified:

- Many models ~100
- Different kinds of models, e.g. environment, mechanics, software
- Variants of the systems that a model represents/simulates
- Versions of a model, e.g. due to error correction
- Different sets of system parameters for parametric models
- Different operating systems and/or computer platforms for simulation execution
- Variants of "the same" model, e.g. different levels of fidelity/accuracy for use in different simulation environments (with different computing capabilities)

There are also challenges due to introduction of various MBSE (Model Based Systems Engineering) methods and tools which provides a range of model types, notations, and languages [3]. These "new" models together with "old" or "legacy code" models have to be integrated in some way. This aspect of the problem is not further described here.

2 Configurations and usage of models

Simulations at aircraft level made from a set of integrated subsystem models are used in different contexts:

Early validation - Simulations support the description and common understanding of new ideas and functionality offered to potential or existing customers. Flexibility in the model world ("the virtual aircraft") enables rapid prototyping and evaluation of various concepts.

Design - Involves both exploration of the design space and investigation of errors and bugs in early stages of development

System verification - In the case of safety-critical systems it is crucial to verify functionality in a simulated environment before real usage in order to reduce risk or even to be allowed to perform "first flight" of a new configuration at all. Verification includes troubleshooting of unwanted behavior or performance (bugs/errors).

Training - Simulation models are built into training products/simulators for both pilot operations (Mission Training) and ground crew operations (Maintenance Training).

Fig. 2 gives an overview of the combinatory problem: Product variants (aircrafts) simulated in various simulation systems. Some combinations are not applicable.

Product Variant	No of seats	Develop. design	Develop. validation	System verification	Ground crew training	Pilot training	**Main purpose**
		Desktop simulation	Laboratory simulation	System simul. Incl. ECU h/w	Maintenance Training	Mission Training	**Simulation type**
Test A/C 1	1		N/A		N/A	N/A	
Test A/C 2	2		N/A		N/A	N/A	
A/C Var A	1						
A/C Var B	2						
A/C Var C	1						
A/C Var D	2						

Fig. 2. Variant matrix, product variants versus simulation type

The Product Line approach enables the model assets to be configured to represent different product variants but also configured for the different usage contexts described above.

3 Information models

Several information models related to product data, product variants and model/software configuration management are studied. The fundamental terminology in this work regarding configuration and data management is based on [5] and [7], and regarding product line development on [1], [6], [12] and [14].

3.1 Definitions and nomenclature

This section describes definitions and concepts related to product line development.

Product Line / Family and Variants. A group of individual products may be thought of as an evolving family of products (a product line) that are derived from a common platform but nonetheless with specific features/functions. Each individual within a product line is called an instance or Product Variant (PV). A product line targets a certain market segment and a PV addresses a specific subset of customer needs in the market segment. All PVs share some common structures and product technologies, which form the platform of the product line. There are three important aspects of a product platform:

- Its modular architecture
- The modules' interfaces
- Rules to which the modules must conform

Product Configuration from Modules. A modular platform is used to create product variants through configuration of already existing modules. The product line architecture enables/restricts the arrangement of the functional elements into modules and the way in which these modules interact [15]. There is a trade-off between modularity and integrality. In simulation architectures the optimal trade-off is different for the different simulation contexts: validation, design, verification and training.

Design for Variety. Product variety is the diversity of products that is provided. There are two main types of variety:

- Technical variety
- Functional variety

Technical variety is relevant to development, testing, and production and is focused on actually reducing the technical variety to gain cost advantages. This includes activities such as variety reduction programs, functional sharing, and design for modularity.

Functional variety is related to customer satisfaction and aims at increasing the functional variety and is more focused on development of the market and business plans, such as product line structuring, product positioning and so on.

A **Product Family Model** contains a definition of the whole family, as well as of how any member of the family can be specified. When the configuration of variants is in focus, it may also be called a configurable product family model [12]. One important property in product family design (compared to that of a single product) is the simultaneous handling of multiple products.

The **Sales Configuration** discipline deals with product customization that supports fast and reliable specification of product variants. Handling of the rules for variability and configuration/integration is strong and relies on knowledge about connectivity implemented in a configurator tool/system. Features of sales configurators are very similar to the needs of "simulation configurators". Methods for analysis and definition of a product range and the standard product [10] will also apply for the simulation model set.

Software Product Line (SPL) development is a way of organizing software and its release structure, similar to the Product Line approach described above. Key interconnected SPL activities are:

- Core asset development (e.g. the models of various aircraft subsystems/equipments)
- Product development (integration, build and delivery of simulation systems)
- Management (planning for updates, delivery, resource allocation, etc.)

Quotation from [6]: *"In a product line effort, Configuration Management (CM) is more complicated and reaches across all of the core asset and product-building projects and possibly even across product lines. It is usually appropriate to plan configuration management at the organizational level."*

3.2 Configuration and data management

Configuration models are used for handling data for development, certification, delivery, and

maintenance in a PDM/PLM (Product Lifecycle Management) context.

There is no common industrial information model for data stored and maintained in CM or PDM/PLM systems, but information about and relations between the PDM objects may be defined as in [4]. Data included in a PDM system typically consist of

- Core product information objects:
  - Configuration structure
  - Specifying and reporting documents
  - Realizations of various types
  - Product configurations
- Planning objects:
  - Development steps/increments
  - Product features and functions
  - Work Packages – work items
- Change Control objects:
  - Change Request (CR) feature/function
  - CR breakdown structure
  - CR responsibility, definition of work

Product parts are handled in traditional PDM/PLM systems and means to also manage models of those parts is to some extent supported. There is, however, information missing in general PDM systems to fully support the configuration activity for simulation systems. For example, models representing the aircraft physical environment are not part of the product definition. This is true for models representing atmosphere, wind, vortex, and runway locations but also for tactical scenarios, pilot behavior and similar.

Modeling and simulation environments, on the other hand, typically provide interfaces for integration with standard software configuration management (SCM) tools. Such tools are inherently strong in software version management, but lack support for model version and variant handling [2]. Model development environments need parts of the integrated support found in PDM systems for evolution and maintenance of realized products. Even if configuration of simulation model integration is not straightforward with PDM methods, change management (identification and definition of changes of models) aligns well. To manage constraints (rules) in the combinational logics of models, the sales configuration domain is strong and may support configuration of simulation systems.

3.3 Parametric models

Due to their flexibility, parametric models are particularly useful in modular simulation products, but also during the different product development stages. This means that parametric models fit well into an SPL approach. Definitions of different parameter types are illustrated in Fig. 3.

Configuration Parameters (CP) are created through the PDM system and used during the simulator build/integration steps and prior to a simulation run to instantiate the models with a consistent set of System Parameters (SP).

In PDM terms the model can be viewed as yet another realization of the PDM part.

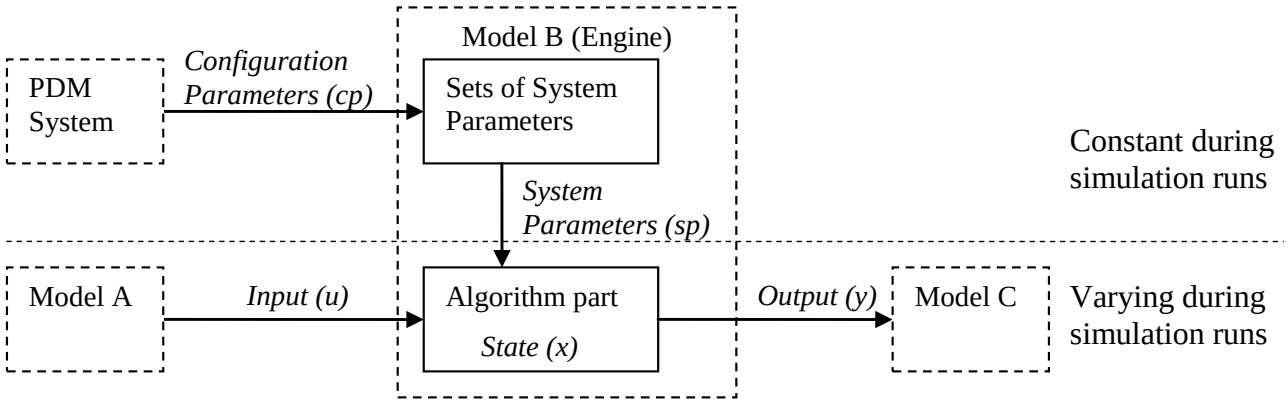

Fig. 3. Model interactions: Input, Output, State, System Parameters and Configuration Parameters used in parametric models.

Typically, the CP are stored in PDM while SP are handled in a SCM tool, and in this work a proposed connection between CP and SP on the conceptual level is presented.

3.4 Variant Configuration Language

There have been attempts to support configuration activities with specialized tools and languages [11], [16], whereof one language is briefly described here. The XML-based Variant Configuration Language, XVCL, is a general-purpose mark-up language for configuring variants in a variety of software assets such as software architecture, program code, or test cases. In [11] four aspects are described:

- To cope with a large number of features and feature dependencies
- Functional dependencies among features implying that only certain combinations of variants may coexist
- Adding new features to make new releases across all existing system variants (past releases in operation)
- Selective propagation of new features (or other changes, error correction) to past releases

3.5 Configuration data, CNA

To describe product configuration data the similarity to genomes in biology may be used. Assume that every product instance (car, truck, aircraft) is viewed as an individual with its own specific configuration (genome), and that the configuration data is possible to represent in a structured information storage object. Let's call this object a CNA-string (ConfigurationN datA string). The CNA is used for on-board configuration i.e. loading a specific configuration into the computer/avionics system of a vehicle prior to usage, in order for its embedded system to "know" what configuration it has. It is also used for On-Ground configuration, i.e. loading a specific configuration into a simulator and setting parameters of included models to consistent values. Examples of data in some elements of the CNA-string are shown in Fig. 4.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--****************************************** -->
<!--* CNA string                             -->
<!--* for Product Configuration 1234567      -->
<!--* Generated date 2010-06-22              -->
<!--****************************************** -->

<CNA name = "spec 1234567" version = "1.1">
  <category name = "CP"
            description = "Config Param">
    <parameters>
      <engine_type part_no = "302"/>
      <gps_installed value = "true"/>
    </parameters>
  </category>

  <category name = "SP"
            description = "System Param">
    <parameters>
      <empty_w weight = "5630" unit = "kg"/>
      <veihcle_l length = "14.2" unit = "m"/>
    </parameters>
  </category>
</CNA>
```

Fig. 4. Example of a CNA-string in XML format.

By using the CNA approach, a tool for cross-referencing or mapping of CP onto SP is introduced. The various validated CNA-strings containing consistent configurations can be stored in the simulation environment for easy access. An extension is to also store a consistent input and state vector for reliable initialization of the simulation model(s).

4 Application example

The application example is a product line effort for the Gripen fighter aircraft simulation models and examples of how they are integrated and used. See further [3].

4.1 PL approach

As some models in the model store are used for several aerospace products the assets (models) reaches across product lines. Examples are the atmosphere, and wind models which are used for all airborne product lines (e.g. UAVs, fighter aircraft, space vehicles) at an enterprise level. The study is however limited to the Gripen product line, and all models are assumed to be fully managed within that scope.

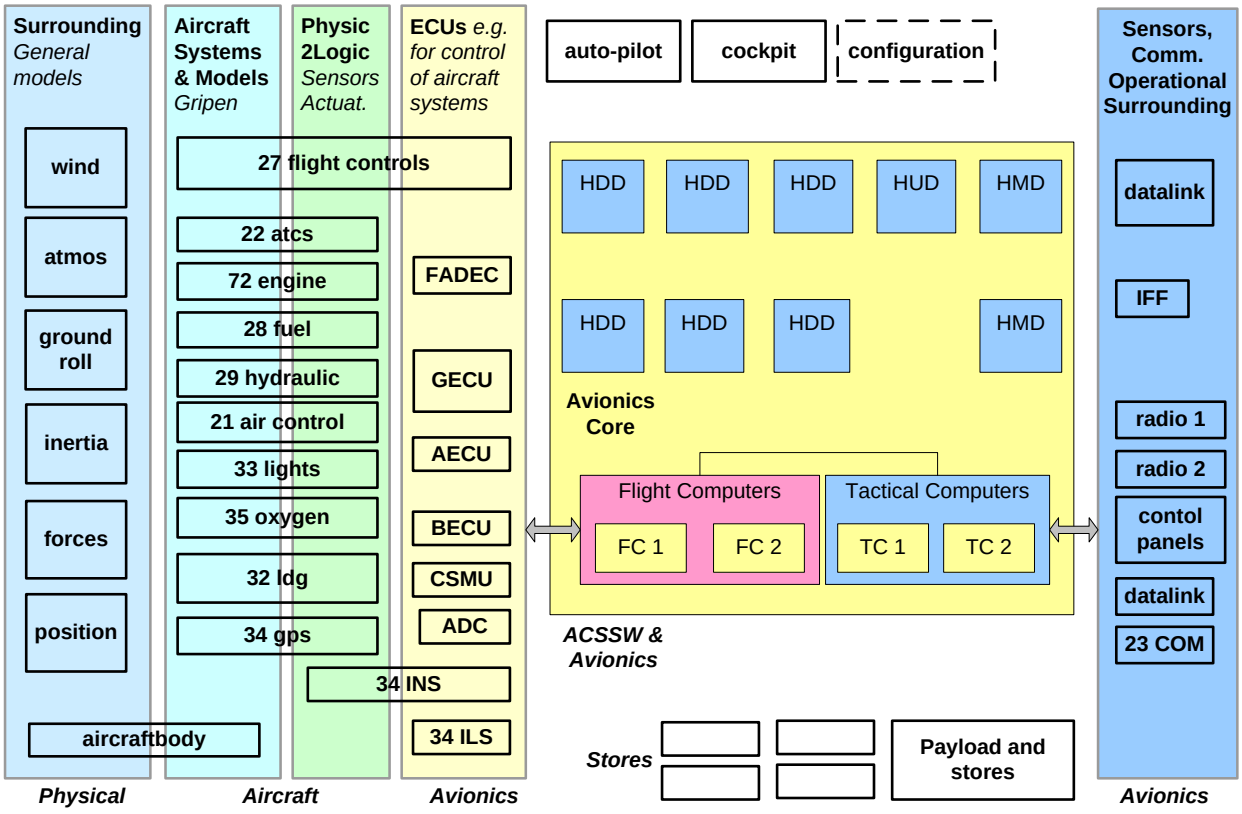

Fig. 5. The model map shows a simplified model overview of the application example.

4.2 Included models

During the initial analysis step of the project a model overview was developed to gain a common understanding of included models and how the models are classified. The following classes of models (in a CM context) are identified, with reference to the simplified version of the overview in Fig. 5:

- Models of aircraft surrounding, classified as "Physical" in the figure, are normally not included in the product part structure. Example: the atmosphere model.
- Models of aircraft parts (mechanical assemblies or whole aircraft subsystems, but also sensors and actuators) classified as "Aircraft". Example: the engine model.
- Models of avionics product parts (e.g. electronic control units, ECUs) classified as "Avionics". Example: the Air Data Computer, ADC.
- Models of embedded software classified as "ACSSW" (Aircraft Computer System SoftWare). Example: the navigation software.
- Models of payload and stores that may not be included in the aircraft product structure but are separate products classified as "Stores". Example: the droptanks.

There are also some special models, for example: pilot behavior and tactical scenarios.

4.3 Use Case examples

Simulation modeling, including scenario creation, is considered to be a basis of general simulation frameworks [8]. Batch simulation runs are typical scenario examples of simulator use. They are relevant for development and verification contexts, but not for training.

To be able to repeat simulations over and over again a set of reusable Operational Usage Scenarios (OUS) are created and stored in a scenario library. An OUS is based on a pre-condition where the aircraft and its systems are "parked" in a steady state or stationary operating point, on the ground or in the air.

From this steady state a dynamic evolution/simulation is started.

Basic scenarios are typical "sunny-day" usage of the product, with all systems fully functional. It is possible to introduce failure(s) as a pre-condition in the stationary operating point or during the dynamic simulation. The time-efficiency of the simulation runs depends on the time it takes to obtain a stable operating point in advance of each simulation. One way to increase the efficiency is to calculate and save steady-state solutions to the library in advance and use these to initiate the simulation model during the batch run.

Fig. 6 gives an introduction to three use case variants, how batch mode simulations are structured when adding new product variants, when adding new specified usage, or when re-testing existing definitions. Given

- a set of Product Variants (PV)
- a set of Operational Usage Scenarios (OUS)

there is a matrix with mappings of applicable OUS versus PV as shown in Fig 6.

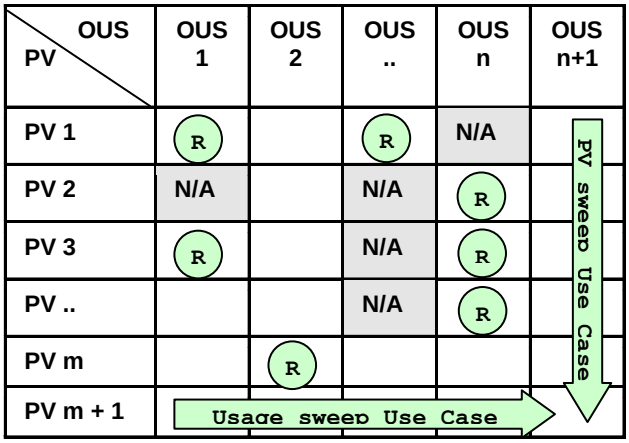

Fig. 6. The matrix shows operational usage scenarios to be simulated for a range of product variants. Regression test points are also indicated.

4.4 Usage Sweep Use Case

When a new product configuration or variant is decided to be offered or delivered, simulations need to be performed for applicable OUS. In batch script pseudo-code it will look like:

```
With selected PV [m+1]
  Integrate models
  Instantiate the models
  For all OUS [1..n+1]
    Initiate the models
    Simulate PV for given OUS
    Store results for evaluation
  End for
End with
```

This simulation pattern is typically relevant when a PV is selected for production and will undergo test and verification activities as basis for certification and delivery.

4.5 PV Sweep Use Case

This Use Case variant defines a need from the opposite viewpoint. Simulation is performed for a OUS for all applicable PVs. The corresponding pseudo-code is:

```
With selected OUS [n+1]
  For all PV [1..m+1]
    Integrate current PV models
    Instantiate the models
    Initiate the models
    Simulate PV for given OUS
    Store results for evaluation
  End for
End with
```

Thus, the challenge in the second use case example is to build, instantiate, and initiate a set of consistent simulation models that are representing (all) the product variants within the product range. The Use Case is relevant when a different usage of the product (family) is defined, e.g. a new defined combination of internal/external stores, or operation in "new" environments.

4.6 Regression test Use Case

The third example applies to modifications or corrections in subsystems of the product line (including existing products). In order to ensure functionality and performance to some extent without analyzing an overwhelming amount of simulation results, a subset of indices of the PV-OUS matrix is pre-selected for thorough regression testing. In pseudo-code format:

```
For all selected PV-OUS indices
  Integrate current PV models
  Instantiate the models
  Initiate the models
  Simulate PV for given OUS
  Store results for evaluation
End for
```

Selected indices are viewed as circles with an "R" in Fig. 6 specifying regression test PV-OUS pairs. Regression test examples could be e.g. changed sub-supplier of some aircraft equipment.

5 Results

Results are gained in the investigation of how the PDM structure and the simulation model structure are aligned and also in modeling of the simulation models variability through a Product Variant Master.

5.1 Alignment between PDM and simulators.

Preliminary results show that the data needed to configure an aircraft simulation can only partly be obtained from the PDM data-set because there are models representing things not included as parts in the aircraft part structure. Of the different model classes listed in section 4.2, the "Aircraft" and "Avionics" classes are well aligned. The "ACSSW" and "Stores" classes may be aligned depending on the level of details handled in the system. The models in the "Physical" class have to be treated as separate parts if the PDM approach should provide any value.

Further, it should be noted that the simulation model structure does not align to the (PDM) product configuration structure, due to:

- separation of the algorithm part from the data part in parametric models
- models are clustered into bigger models representing several parts in the PDM part structure.

The two different structure patterns are illustrated in Fig. 7 and 8 respectively.

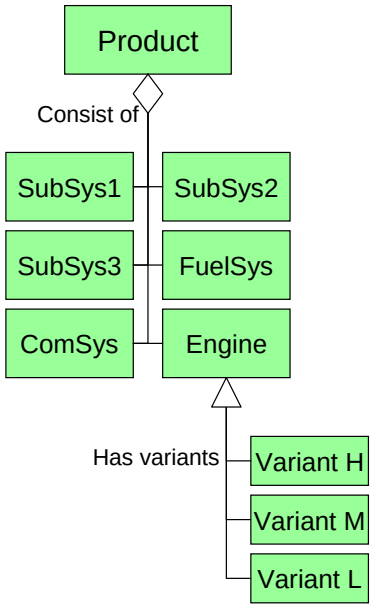

Fig. 7. Product part structure.

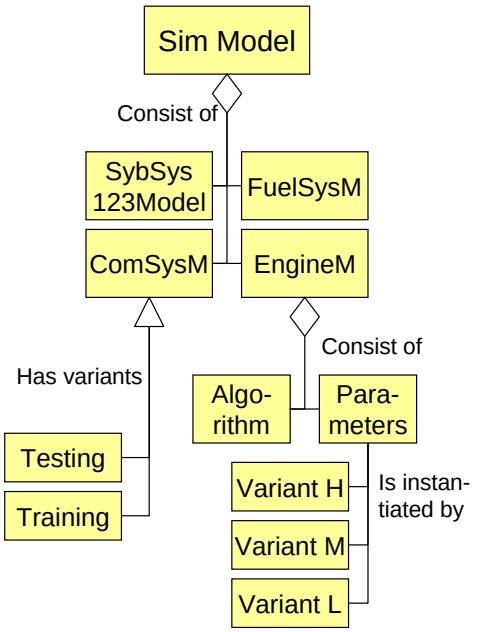

Fig. 8. Model structure.

Change management through PDM concepts is shown to be usable for models, even though the models themselves are not included in the PDM system. Requests for and planning of model changes are defined through Change Control and Planning objects. Configuration of models is better suited to be performed in the simulation environment.

5.2 Product Variant Master

The product line model is partly implemented as a "Product Variant Master" PVM, according to the methodology in [9].

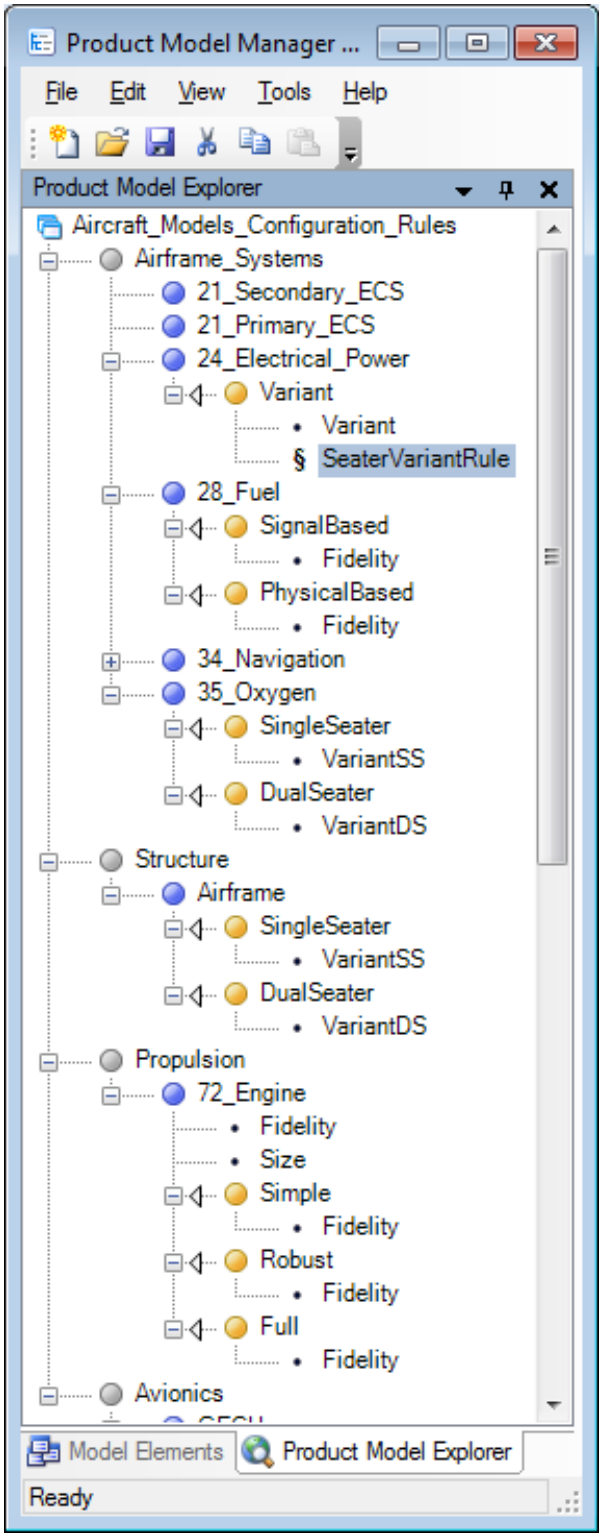

Fig. 9. Product Variant Master model for the application example.

The PVM includes rules to constrain the combinations in order to avoid a combinatorial explosion. A screenshot from an example PVM model in the Product Model Manager tool [13] is shown in Fig. 9.

6 Conclusions and further work

The following conclusions have been drawn so far in the project;

- As computer performance increases, the number of useful simulation models representing products and/or product parts grows in industry and the models are also included in training of end users.
- There is a need to support the configuration specification activity in order to integrate and build consistent combinations of models into large-scale simulations.
- A method based on genealogy principles for specifying a product's configuration in a so called CNA-string is introduced.
- The simulation model structure does not align to the (PDM) product configuration structure due to separation of algorithms from data and that models are clustered to represent several PDM parts.
- The CNA-string approach is proven to work in practice and is promising as a basis for integration of configuration data between the PDM and simulation tools.

Further work is to develop methods to support the build/integration process steps from configuration specification to back-reporting of actual simulation configuration. Comparison methods are needed to analyze differences in specified configuration versus actual configuration. More stringent mapping of product models by meta-modeling is planned.

9 Acknowledgement

The authors wish to thank their supervisors Petter Krus at Linköping University and Erik Herzog at Saab Aerosystems, for their support. Funding for this work was provided by the Swedish National Aeronautics Research Programme, NFFP5.

References

[1] Alizon F, Khadke K, Thevenot HJ, Gershenson JK, Marion TJ, Shooter SB and Simpson TW. Frameworks for product family design and development, *Concurrent Engineering Research and Applications*, 2007., pp. 187-199.

[2] Altmanninger K, Brosch P, Kappel G, Langer P, Seidl M, Wieland K, and Wimmer M, Why Model Versioning Research is Needed!? - An Experience Report. *Proceedings of the Joint MoDSE-MC-CM 2009 Workshop*. Denver, 2009

[3] Andersson H. Aircraft Systems Modeling - Model Based Systems Engineering in Avionics Design and Aircraft Simulation. *Licentiate Thesis, Linköpings universitet.* Linköping, 2009

[4] Andersson H. Variability and Configuration Principles for Simulation Models in Product Line Development. *7th European Systems Engineering Conference EuSEC*. Stockholm, 2010.

[5] ANSI/EIA-649-A. National Consensus Standard for Configuration Management. 2004

[6] Clements P, Northrop L. Software product lines-practices and patterns. *Addison-Wesley*. Boston, 2002, pp 303.

[7] Dahlqvist AP, et al. PDM and SCM - similarities and differences. *The Association of Swedish Engineering Industries*. 2001

[8] Harrison, N., Gilbert, B., Jeffrey, A., Lauzon, M. & Lestage, R. (2004). Adaptive and Modular M&S Configuration for Increased Reusability. *Interservice/Industry Training, Simulation and Education Conference*. Orlando, 2004.

[9] Haug A, Hvam L, Mortensen NH, Lundvald S and Holt P. Implementation of conceptual product models into configurators: From months to minutes. *5th World Conference on Mass Customization & Personalization*. Helsinki, 2009.

[10] Hvam L, Mortensen NH and Riis J. Product Customization. *Springer-Verlag*. Heidelberg, Berlin, 2008.

[11] Jarzabek, S. Effective Software Maintenance and Evolution: Reused-based Approach, *Auerbach, CRC Press, Taylor and Francis Group*, New York 2007.

[12] Krueger CW. Towards a Taxonomy for Software Product Lines BigLever Software, Inc., USA,

[13] PMM, Product Model Manager, *Incore Systems*, www.incoresystems.dk, Retrieved on (2010-06-02).

[14] Sivard G. A Generic Information Platform for Product Families. *Doctoral Thesis, Royal Institute of Technology*. ISSN 1650-1888, Stockholm 2001.

[15] Ulrich K and Eppinger SD, Product Design and Development. *McGraw-Hill*. New York, 2008.

[16] Jiang P, Mair Q, Newman J, and Huang J. UML and XML based change process and data model definition for product evolution, *Software Evolution with UML and XML, H.J. Yang (ed.), IDEA Group Publishing*, pp. 190-221, 2005.

Copyright Statement