Enhanced model-based engineering for centrally managed configuration management in product lifecycle management

Christian Deschner[1*]

[1] Siemens Healthcare GmbH, 91058 Erlangen, Germany

Abstract. In times products gain in complexity and variety whereby release and development cycles become even shorter, consistent and systematic variant management is essential not only for technical communication but also for the very most processes in PLM. Therefore, system engineering and system configuration themselves must be leading for a centrally managed, reliable variant management for all PLM processes. We depict how enhanced model-based system engineering approach based on product and product component models can be the enabler for variant management in all PLM processes by specific, explicitly deduced views in different stages of the entire Product Lifecycle.

1 Introduction

In the very beginning of industrial car manufacturing, Henry Ford allegedly said, customers can buy the famous T-model in "any color so long as it is black". This statement clearly depicts how variant management was handled in former times, where cars have been pure pieces of hardware, assembled within factory as a monolithic good.

Since then, modern products have become more and more complex and diverse. Vertical range of manufacturing has decreased, and software has become even more important.

This – still ongoing – process makes modern products more and more complex and built-up from numerous single sub-components manufactured, engineered and integrated asynchronously widespread all over the world.

Therefore, integrating all sub-components and its information to real, sellable products is one of the most challenging aspects in modern product lifecycle management processes. In order to manage this challenge systematically, all relevant product engineering and configuration information is systematically accessible at each stage just in time.

2 Multilevel component-based system engineering

In times of IoT, increasing individualization, and even shorter development cycles, products and product lifecycle management (PLM) processes and correspondingly all product integration and product information integration processes are changing.

Smart and high-tech products nowadays consist of hardware and software. Modern medical devices like CT scanners, smart home solutions and modern cars are a symbiosis of hardware and software.

Both, hardware and software very often are also no monolithic components, but are assemblies consisting of different subcomponents with independent lifecycles.

Building up products means, that different assemblies, assembly groups, accessories, software libraries and applications are integrated, combined and configured bottom up in multiple stages as multilevel assemblies.

In terms of software, a commercial or open source operating system is used and maybe configured for individual purposes. Different third-party software and/or self-developed applications are to be run on the operating system, integrated and configured specifically. The same holds true for hardware, where different assemblies and parts are to be combined. Gearboxes, motors and chassis of cars as well as gantries, housings, power supplies and patient tables of Computed Tomography (CT) scanners.

At the latest when hardware and software components are integrated, there will be a clash of domain cultures. More traditional hardware development setups meet agile software development approaches with different tools and processes. Nevertheless, both worlds can't be treated as isolated domains, since there are functional dependencies between different subcomponents and groups. A patient table of CT scanner can be controlled by software from the workstation or a tablet, as well as by hardware like joysticks or buttons.

Depending on product types, portfolio definitions and strategies, these multilevel assemblies might not be sellable products, but platforms for product families or mass customization and define large numbers of valid product configurations. In mathematical terms, the assemblies can be conceived as complex formulas with given value sets defining logical solution sets. That

* Email: christian.deschner@siemens-healthineers.com

means, a specific product can be derived from the overarching product platform or family by defining specific configuration settings and selections for all relevant parameter and options. Since any configuration setting and selection may affect different subcomponents (assembly groups, software features etc.) within the entire assembly, all relevant dependencies, parameters, options and constraints within and/or between subcomponents must be well defined and systematically evaluable.

2.1 Model based system engineering and model driven software development

A well-established methodology in hardware engineering is the "Model Based System Engineering" (MBSE) approach which is defined as follows:

> *Model Based System Engineering (MBSE) is "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [1].*

Typically, with this approach, all technical constraints and within one hardware assembly are systematically created during the engineering process.

For similar methodology for software development, is the "Model Driven Software Development" (MDSD) approach. MDSD is designed to automatedly create software by using standardized modelling languages, code generators and interpreters.

If only MBSE and MDSD are applied consistently, all technical constraints, dependencies, configuration items and interfaces will be well defined for resulting engineering artifacts and product subcomponents.

Even if these engineering methodologies are internally applied for all subcomponents, there is a gap for the multilevel assembly scenario. Whereas in software engineering, APIs usually are well documented, MBSE mostly focusses to the inner structure of the engineering artifact only, but not systematically to its usage or configuration within other, higher level artifacts.

Consequently, the missing links for systematic, multilevel product engineering are:

- Hardware API documentation standards and interfaces in MBSE
- Compatibility of these standards with to software API standards
- Enhancements in software API standards for hardware related settings and constraints
- Standardized and precise set of meta information

By closing these gaps, all engineering and configuration artifacts created asynchronously and independently can be integrated to consistent and comprehensive models. This enhanced Model Based System Engineering approach bridges bottom-up system development and top-down product and variant configuration.

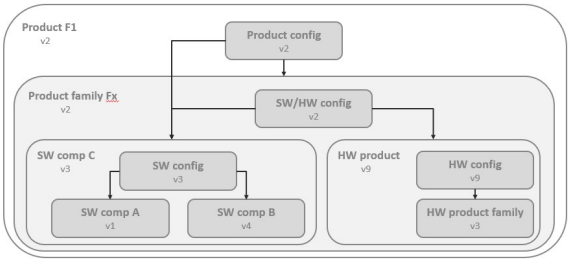

Fig.1. Multi-level component-based system and product assembly

2.2 Benefits of enhanced model-based system engineering

In agile and short-term development setups, product and product information integration is one of the most challenging tasks in many process steps. New components and updates must be systematically integrated just-in-time, including a reliable and comprehensive impact and risk analysis for the entire system.

Any change in system or product configuration may have impact in different aspects all over entire product related processes. It's not only about technical constraints between different components and subsequent updates and changes. Replacement only one small component in a hardware assembly might have strong impact due to embargo regulations. Replacing or updating an open source software component might result in obligations to disclose sensitive data and knowledge.

These examples show, how much different information, data and constraints need to be integrated into entire "Digital Twin" models of complex products and systems. Well defined and documented data structures and appropriate tools are necessary to handle this complex setup and to make any kind of relevant data and information generally accessible.

3 Unique models and specific views

The resulting complex and big data structures with multidimensional dependencies, properties result in big ontologies as single access points to any kind of engineering related information. These data structures must be unique and as exact as possible within its scope**.**

As a bold and simple postulate, the model shall be trusted as the truth.

In mass customization or product family setups, concrete product configurations and/or variants are created by filter mechanisms and configuration settings applied to a master model for product family models. Resulting product models are real subsets of the basic master model with a reduced scope but with the same information and data space.

3.1 Specifically configured product models

To preserve the information space and to ensure completeness and correctness of the resulting product instance models, configuration settings must be applied to the master model in multiple steps by following a from inside (in model engineering settings) to outside (outer configuration setting from higher levels) and bottom-up approach.

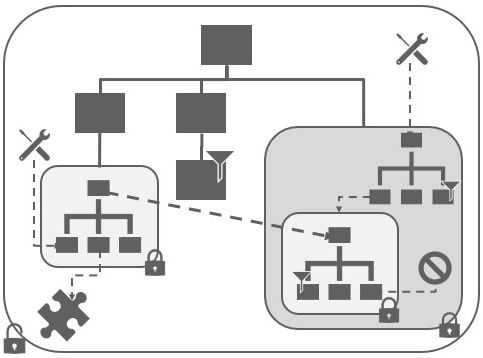

Fig. 2. Deriving a specific product model from a product family model in a multi-level engineering setup.

The process how to create a product variant from product family (master) can be described as follows:

1. Resolve engineering settings per model
   a. Remove "filtered out by validity element" elements
   b. Remove elements by essentially depending "dead" relations (relation types) after step 1.a.

2. Resolve configuration settings bottom up (component to model)
   a. Remove "deactivated by config" elements
   b. Remove elements by essentially depending "dead" cross model relations (relation types) after step 2.a.

3 Specific views based on unique models

Based on engineering models, specific views for specific purposes and sub-processes can be extracted. These views are specific aspects to the entire system and reduces the complexity to the minimum for a given purpose.

In contrast to the uniqueness and absolutely correctness of the model, specific views can't be declared neither as unique nor as absolutely correct but need to be stable and sufficient for its designated purpose. Views are of reduced complexity and carry only a subset of the entire information from the base model. Whereas systems models result in an ontology-based data structure, views most likely will end up in more simple taxonomies.

From a processing point of view, it's very important, that underlying qualitative and quantitative constraints and considerations for the specific purposes can be well defined and implemented in a ruleset on how to automatically derive the respective view in a suitable data format. Examples for such specific views and formats are system test plans to be imported to a given test management system, publication structures imported to Content Management Systems and bills of material for SCM tools.

Systematically created views based on stable rulesets applied to specific system models and revisions, complexity can be removed from specific sub-processes, scalability for entire product engineering process can be increased.

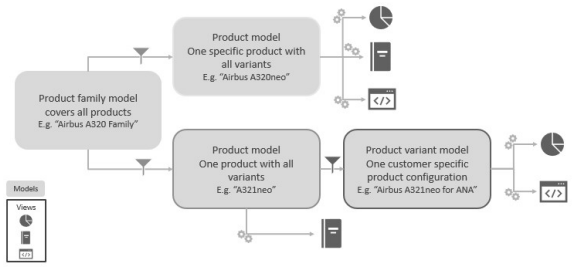

Fig. 3. Deriving specific views from a specific product in a multi-level engineering setup

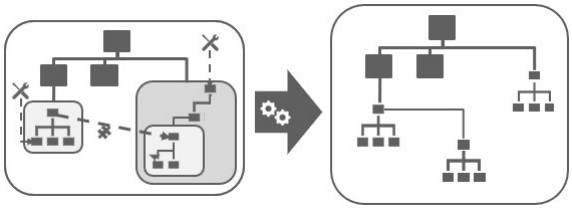

Fig. 4. Deriving a specific view (taxonomy) from a product model (ontology)

Creating specific views from a model also requires a fixed proceeding to ensure well-formed, self-contained and valid output.

The transformation process how to create views from models can be described as follows:

1. Transforming an Ontology Model to a View

   From a defined root element…
   a. Removing elements to be ignored
   b. Removing relations to be ignored
   c. Removing (resulting) dead-end relations
   d. Removing resulting free elements/sub models
   e. Rule based resolving of remaining relations by precedence

2. Transforming Taxonomy to target format
   a. Sorting and nesting structure by explicit criteria
   b. Transformation to specific target format

Whereas this process holds true for all views, rulesets and transformations must be defined for each view specifically depending on purpose and further processing.

Ruleset must contain specific definitions and constraints for the different processing steps.

Rules and constraints for steps 1.a to 1.d:

Ignore elements by

- type
- values
- type and property values

Ignore relations by

- type
- left/right element
- type and left/right element

Rules and constraints for step 1.e:

Precedence of relation type to be processed in a defined a sequence

Rules for relations to be defined by

- type
- left/right element
- type and left/right element

and to be processed like

- Ignore/remove
- Resolving/transformation to parent/child structure
- Multiply to left to right
- Multiply right to left
- Move left to right
- Move right to left

Transformations for steps 2.x:

Rules for rearranging the view (nesting sequencing, moving elements)

Data transformation for specific data format (e.g. XSL)

4 Summary

In this article, the challenges of widespread, agile and component based system and product engineering and its consequences to system integration and configuration are discussed.

To enable different process steps within the entire Product Lifecycle to follow short-term and agile development processes and diverse product portfolios, all relevant engineering and configuration information of all engineering artifacts must be accessible and integrated within all stages of a multilevel system engineering setup.

Subsequently, this article describes how modern development approaches like Model Based System Engineering (hardware) and Model Driven Software Development (software) can be integrated, and therefore be the basis for closing the gap between component-based engineering information and multilevel system and product integration and configuration.

Next to this basic concept, basic considerations to unique component and system models are described, and how specific views are delimited.

Finally, basic rules and constraints necessary for ensuring validity and completeness of specific product, product variants and custom configurations are figured out.

References

1. INCOSE, *Systems Engineering Vision (2020), INCOSE-TP-2004 (September 2007)*
2. Deschner, Ziegler, *Informationsmanagement und Ontologien – entwicklungsnahe Produkt-dokumentation in der Medizintechnik, tcworld conference Stuttgart 2018 (November 2018)*