Collaborative Development of a Space System Simulation Model

Volker Schaus*, Karsten Großekatthöfer**, Daniel Lüdtke*, Andreas Gerndt*
*German Aerospace Center (DLR)
Simulation and Software Technology
Software for Space Systems and Interactive Visualization
Lilienthalplatz 7, 38108 Braunschweig, Germany
Email: {volker.schaus, daniel.luedtke, andreas.gerndt}@dlr.de
**Astro- und Feinwerktechnik Adlershof GmbH (AFW)
Albert-Einstein-Str. 12, 12489 Berlin
Email: k.grossekatthoefer@astrofein.com

***Abstract*—Modeling and simulation is a powerful method to evaluate the design of a space system. Simulation models represent valuable knowledge and require considerable time and effort for their development. Means for reuse should be taken into account from the beginning of model creation. This paper presents a collaborative model development process, which creates prerequisite information for successful reuse of simulation models. It introduces a knowledge model and proposes reviewed documentation at each step in the process. These pieces of documentation enable successive reuse at different levels. The modeling process was evaluated by creating a system simulation of the OOV-TET-1 satellite including three satellite subsystems, dynamics, kinematics, and space environment. Furthermore, the organization of models and their documentation artifacts is crucial in order to search, find, and reuse models across project partners and across projects. The paper suggests a flexible model database that suits the special requirements of typical space projects and large research organizations.**

***Keywords*—model development process, model reuse, model database, space system simulation**

I. Introduction

Space systems tend to be complex system designs with many interdependencies between various subsystems, components, and the payload. Typically, the development process lasts several years, involves an interdisciplinary design team, and follows the well-established Systems Engineering V-model [1] that has been tailored to the space domain [2][3].

Modeling and simulation has become an inherent part of the design process. Simulations are mainly used during the domain specific detailed design. Basically, all engineering disciplines have developed special computer-aided engineering tools that are specific to their domain. For example, control engineers model the attitude and orbit control system (AOCS) of a spacecraft and use a continuous or discrete simulation to validate orbit maneuvers and attitude control performance. One of the standard software packages used in this field is MATLAB/Simulink. On the other hand, thermal engineers set up a simulation based on a geometry model, the orbit, the exposure to direct sunlight, and internal heat sources that calculates the developing of thermal housekeeping data and temperatures of satellite components. Widespread tools for thermal analysis are ESARAD/ESATAN or ThermalDesktop. Such examples can be named for all engineering domains involved in space system design. However, all these simulations stay within their domain. Portability and data exchange between the simulations are major issues and make the evaluation of cross-domain dependencies very difficult. Recent initiatives, such as European Space Agency's (ESA) SimVis project [4] and German Aerospace Center's (DLR) Virtual Satellite initiative [5] aim to establish overall system simulations during the early system design phase in order to evaluate system performance.

Within a project, collaboration among the design team is one of the key factors for a successful design. In this context, collaboration usually stands for teamwork, communication, sharing information, and product data exchange. Over the past decade, concurrent engineering techniques have been put into practice and special facilities for concurrent design have been established at many sites [6][7][8]. Data exchange and data consistency issues are addressed by defining a global system model that is holding all design parameters. This ensures that the design team always works with the same common data basis and creates a certain awareness of changes and their consequences to other domains or the whole system. Recent attempts are to use this system model not only in early designs but also in subsequent phases, possibly throughout the whole design process [9][10]. This provides consistency and reuse of design information and removes the existing barriers between distinct design phases.

However, teamwork within a project or in a design team is only one aspect of collaboration. For the DLR as a large research organization and satellite designer, two more ways of collaboration are very important:

- model sharing and reuse across projects
- collaboration across partners

Many projects have similar requirements for simulation models; therefore reuse is desired to shorten design time and should be easily possible. According to Pidd [11], reuse strate-

gies require certain features that can be arranged in four categories: abstraction, selection, specialization and integration. An abstract, high-level description is needed to understand the model quickly. Means of searching are necessary to find and compare models and select the right artifacts for reuse. This also requires a certain organization of models, such as a model library or database. Modularity and small granularity is needed to support specialization and to enable the creation of model variants. Interfaces need to be well defined to allow integration.

The common situation, however, is that design time in projects is limited and next deadlines are pushing for fast results. In consequence, model development often means directly building running models as fast as possible. Mostly models are re-implemented from scratch or based on former implementations (code scavenging). There are no common guidelines for developing and documenting models. In addition, existing models are either saved on shared network drives or packaged as model libraries. They are typically developed by working groups and the knowledge is often kept within departments, institutes, and companies. Sharing and searching for potentially reusable models is not easily possible.

The second item regarding the collaboration across partners addresses the fact that DLR in their satellite projects always works together with partners, among them many small and medium-sized enterprises (SME). Typically, large organizations have special departments dedicated to modeling and simulation, which makes the use of simulations cost-effective. However, SMEs face high cost for software licenses and trainings in order to provide simulation models. Again, the issues here are model portability and transformations. The idea is that in the future SMEs can provide a model together with their individual hardware component or subsystem. Such a model can then be integrated into a bigger simulation set up by a simulation department of a larger organization. This situation is especially true for DLR and Astro- und Feinwerktechnik Adlershof GmbH (AFW) who have worked together on several space projects over the years [12][13]. The collaboration was continued in a nationally funded research project on reusable model development with other industrial partners outside the space domain.

In essence, the research focuses on the desire to harmonize the model definition and to build models that enable the application of reuse strategies, thus, providing the basis for model reuse across projects and collaboration across partners.

The paper is organized as follows. Section II presents a modeling process for developing simulation models and describes a reuse activity. Section III explains the application of the collaborative modeling process based on a satellite bus and evaluates this approach. Section IV deals with establishing a model database. Finally, the paper ends with a conclusion.

II. Model Development Process

A collaborative modeling process was developed to create well-documented models and harmonize model definition. Figure 1 depicts this collaborative modeling process as it was used to create simulation models of the satellite bus. It consists of five steps and is based on the waterfall model in software engineering. Each step generates documentation, but not necessarily a document. Starting with a problem description, the requirements are analyzed first. They state what the model should do, what precision, accuracy or performance is needed, etc. Typically, requirements are written in natural language and can be organized as a list. The next process is the model analysis which includes the relevant physics, a mathematical representation, algorithms, and constraints.

The third step is the abstract design phase. Here, the model's interface, its input and output values are defined. It also includes parameter and quantity definitions and different model states, if necessary. Additionally, the internal architecture of the model is defined in this step. The abstract model can be seen as a platform independent model in terms of model-driven architecture (MDA)[14].

The fourth step in the process is the implementation step and implements the model as source code for a given platform, e.g. C, or in a simulation software package such as MATLAB/Simulink. In addition, this implementation always depends on the current environment and software version. It is a so-called platform specific model, again drawing the parallel to MDA in software engineering.

Finally, validation is a testing step for the model. Tests can include unit tests on software level, type tests for correct size, format and physical units of input and output signals, and expected input/output value pairs.

The whole process identifies two roles or actors: the system engineer and the simulation engineer. These two roles pass through the modeling process in close collaboration. The right hand side of Figure 1 indicates the work distribution between system engineer and simulation engineer. At the beginning, the majority of the work is the system engineer's responsibility.

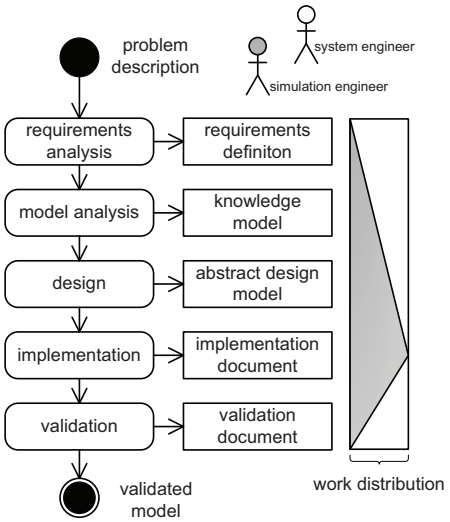

Fig. 1. The model development process consists of five steps, each of them is documented. The work distribution between the involved engineers changes over the steps.

In the following steps, the simulation engineer becomes more and more involved and reaches the peak workload during implementation phases. A review process between simulation engineer and systems engineer assures the quality of the models. Each piece of documentation is reviewed according to the four-eye principle. Once all corrections and modifications are made, the documentation of the current step is released. It is important to mention that the knowledge model plays a crucial role in the collaborative process. It can be seen as the central piece of documentation where system engineer and simulation engineer exchange and discuss information as well as find a shared comprehension of the domain and the model. Of course, several specialist can carry out the roles of system engineer and simulation engineer if the model is more complex.

The five steps of the process can be explained in more detail by introducing a simple example from the space domain: The objective is to model the solar flux for an earth orbit. The Sun's radiation can be described with Stefan-Boltzmann's law. The mathematical equation of the solar flux Φ is dependent on the distance to the sun r:

$$\Phi\left(r\right) = \sigma\, T^4 \left(\frac{R}{r}\right)^2 r \tag{1}$$

In this equation, σ stands for the Stefan-Boltzmann-Constant, T is the temperature of the Sun and R is the radius of the Sun. The assumptions for using this equation are:

- variations and the Sun's cycle are neglected
- Earth's orbit around the Sun is assumed to be circular
- the Sun's radius is assumed to be constant
- the Sun is considered as a black body

All this information is considered as knowledge model. The abstract design model is given in Figure 2 in SysML [15] representation. The interface can be clearly identified: the input is the distance to the Sun r and the output is the radiation Φ. The internal architecture is also easily recognizable: vector lengths and normal directions are calculated first and then Stefan-Boltzmann's law is applied. The implementation in MATLAB/Simulink would look very similar to the abstract model in Figure 2 because Simulink models are also composed of blocks.

During this modeling process, many different artifacts of documentation are created. These artifacts make reuse more convenient or even possible in the first place. The reuse activity diagram is depicted in Figure 3. Basically, it is a bottom-up approach browsing through the documentation artifacts. First, one starts looking at the implemented, validated, and verified model and the attached documentation. If reuse seems feasible at this point, the model can be directly reused. However, this case is very rare. It is very unlikely that all system requirements and platform dependencies match the new working environment. If reuse seems not possible, one can go up one step higher to the abstract model. Here, one checks again if the abstract model is applicable for the new task. If this is the case, reuse of the abstract model is possible and the modeling process is started again at the implementation phase. As a result a variant is created, e.g. the original was in C code and the new one in MATLAB/Simulink. The abstract definition was reused; both models have the same interfaces and internal architecture. If also the abstract definition cannot be reused, one can even go one step higher and look at the knowledge model. Mathematical equations describing the problem, the physics behind the model and certain constraints can be a useful starting point for creating a new abstract model and an implementation.

Finally, if even the requirements change, there is no way around a new development; nevertheless, some artifacts might still be useful. This situation can again be exemplified using the Sun's flux model. One of the constraint mentioned before was that Earth's orbit around the Sun is considered circular. Let's assume there is a request for a more detailed model that also takes the yearly variation due to more precise elliptical Earth orbit into account. The handbook [16] defines a correcting extension to Equation 1 that varies the flux with respect to the day of the year. Of course, adding time as new variant means extending the interface. Different scenarios, e.g. specialization or encapsulation are now possible to create the more detailed Sun's flux model. The developer needs to decide

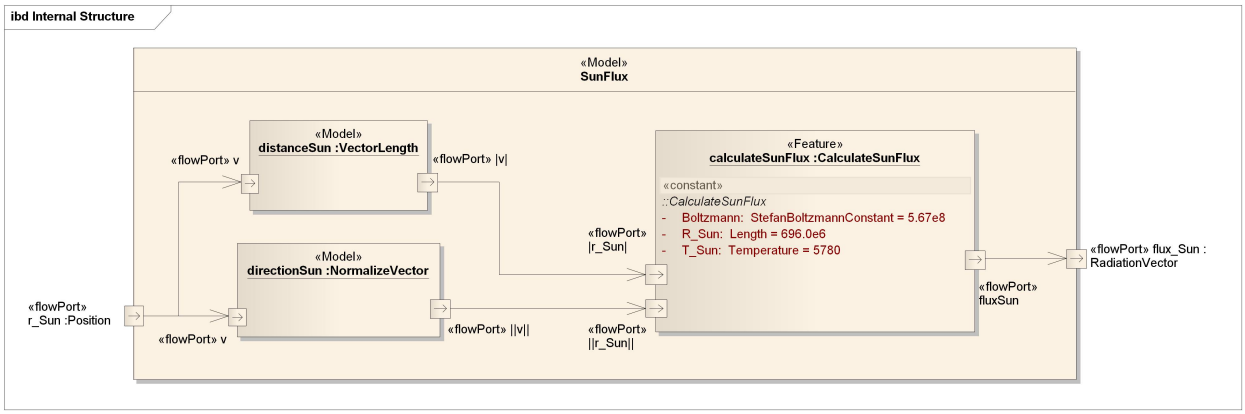

Fig. 2. Example of an abstract model design using SysML: the Sun's flux model with interfaces, internal architecture, and data flow.

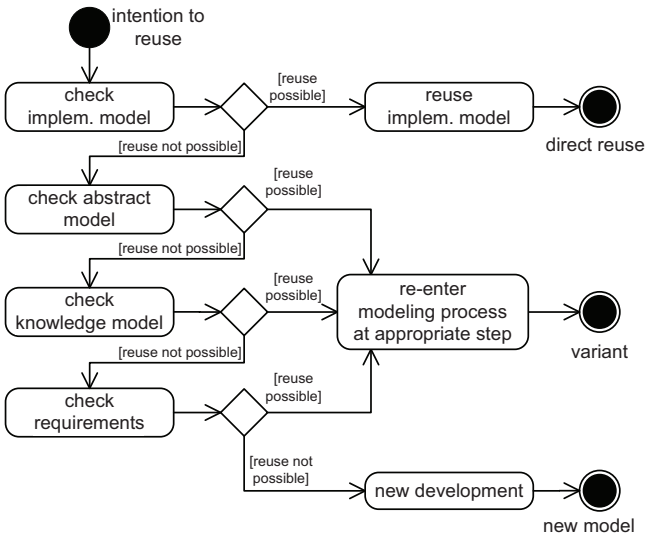

Fig. 3. The reuse activity diagram is a bottom-up approach browsing through the model documentation artifacts. Reuse is possible at every step. Once the appropriate step is found, the modeling process can be re-entered which leads to a model variant.

which method is appropriate. This particular case shows that even if requirements change, knowledge description, abstract design, and implementation of the previous model are still useful to derive a new model.

In summary, this bottom up browsing through the previously created models and their documentation allows flexible and partial reuse at different steps of the modeling process.

III. Process Application and Evaluation

The practical capabilities of the model development process were proved by setting up a dynamic simulation model of the OOV-TET-1 satellite bus system [13] originally developed at AFW. A satellite bus represents the infrastructure of a satellite and thus provides necessary services to operate payload such as measurement instruments or on-board communication antennas. Due to a high complexity involving different physical domains and fields of responsibility, the bus is divided into sub-systems, each fulfilling a special function, e.g. the AOCS determines and controls the satellite's trajectory and orientation in space.

On the top-level view, as shown in Figure 4, the model consists of four major elements: environment, satellite, dynamics, and mission. The environment sub-model considers the satellite's orbit and all relevant space objects and their influences, amongst others the Sun's flux model introduced in Section II. Another sub-model describes the internal configuration and subsystems, in our case power supply, thermal behavior, and attitude control, and thus represents the satellite itself. The dynamics block formulates the interaction between satellite and space environment, i.e. the equations of motion and disturbance torques induced by, for example, the magnetic field of the Earth or the Sun's flux. The mission block introduces the start time of the simulation and defines so the satellite's mission time.

The overall result is a closed loop simulation of the TET-1 satellite bus on the one hand, and on the other hand, a library of more than one hundred models of different complexity and granularity to be used in different phases of future development or research projects. Due to a high degree of modularization, all features and models can be easily recombined to extend the model library and derive model variants of components and subsystems.

Knowledge was captured along the modeling process in different forms of documentation. For example, the knowledge models are actually documents in natural language. The abstract models were described in a specialized modeling software using the System Modeling Language (SysML), which is an extension of the Unified Modeling Language (UML) for systems engineering applications. Within these diagrams, different problem-specific characteristics of modeled quantities such as units or coordinate systems were taken into account by a customized SysML profile. The final implementation was done using MATLAB/Simulink.

Starting from its original state, the satellite bus model was used in different phases of the satellite design process from feasibility studies to integration and test. At AFW, the developed simulation could be successfully used in practice in different use-cases. Particularly, the model was used during the integration and test phase of the TET-1 satellite AOCS [17], where simulated reference data proved to be a valuable support for software tests. Furthermore, a feasibility study to evaluate the evolution of the TET-1 satellite platform also included simulation-based analysis of potential AOCS modifications. At DLR, the knowledge descriptions and abstract models of the space environment where reused for new implementations in Java for an orbit visualization in the Virtual Satellite software framework.

It was found that strictly following the modeling process through all steps is very time-consuming, because the effort of manual editing is considerably high. Consistency between the different development phases and across different models had to be ensured by manual reviews and adaptation because of lacking tool support.

Using SysML for the model design turned out to give a well-defined base for the implementation phase and thus

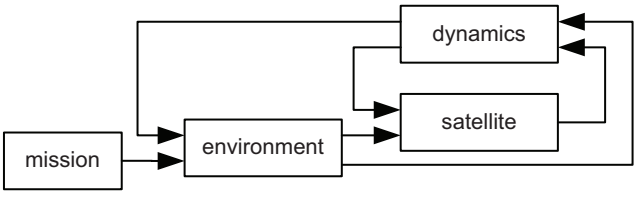

Fig. 4. View of the simulation of the TET-1 satellite bus as it was created in accordance to the modeling process in Section II. Each one of the four top-level blocks is masking sub-models of different complexity and granularity, in total over one hundred individual models.

lead to a notable reduction of the programming effort in this phase. Nevertheless, SysML as a general modeling language for technical systems revealed descriptive limits for the space domain, e.g. interdependencies between different coordinate system or composed quantity types. The incomplete tool integration of the process lead to redundancy of information in the different documentation artifacts. Especially the transformation of artifacts from one process step to the next was intricate, since it involved transformations from one tool to another.

IV. Model Database

To benefit from the development process in the context of model sharing and reuse across projects, some kind of model library or repository is needed to collect simulation models [18]. Such a system would allow a comprehensive overview of available simulation models within an organization like DLR. In combination with the technical infrastructure to collect and catalogue models, common guidelines are provided how simulation models shall be developed, documented, verified and validated. Thus, model reuse becomes much more likely than before.

DLR's project SimMoLib (Simulation Model Library) addresses the goal to build a model library as well as to provide guidelines and best practices for model developers. The process proposed in this paper inspires these guidelines.

The technical requirements for such model library in the context of DLR are quite diverse. On one hand, the library should support a large variety of simulation models for different domains and simulation platforms and, on the other hand, it should ease the process of reuse and integration of models into new projects.

Due to various different simulation platforms and existing tools involved in the space system development process, a tool specific solution, like a library integration into Simulink, is not feasible. Rather the contrary, a simulation and computer platform independent solution is desired.

The library system consists of two logical components: a generic data store and a concept for meta data. The data store allows the archiving of models (source code, binaries, etc.) and additional documents, like the artifacts that are generated in the model development process. With this approach, compatibility with all available modeling tools is guaranteed because model files can be stored on the local file system.

A flexible meta data concept is currently under development to gather additional information and support the search for specific models. This concept is based on so-called profiles. Some profiles collect some fundamental information like model name, short description, key words and other profiles contain simulation tool specific fields like tool version, inputs and outputs, etc.

A particular profile manages the dependencies of a model to other models in the library. For example, the aforementioned sun flux model (Figure 2) depends on two sub models: VectorLength and NormalizeVector. These two sub components are stored as self-contained models in the library and are referenced by the dependency profile. Thus, a download of the sun flux model can trigger the system to provide also the needed sub models.

The meta data profile system allows the management of a large variety of simulation models while maintaining the ability to support simulation tool-specific enhancements. During submission of a new model to the library, a wizard-like system helps the submitter to provide all necessary meta data and combine profiles to match the requirements of a specific project or environment.

Approaches like [19], where an ontology was designed to catalogue the models, seems too laborious for the particular use case in a large research organization due to the large variety of application specific models. Especially, considering that other domains like aeronautics, transport, and energy are also located within DLR, they can also benefit of a common simulation model library. The general idea is to motivate model developers to easily submit their creations to a pool of preserved knowledge without the need to possibly modify or extend a complex ontology. It is assumed that a powerful search capability is sufficient to discover relevant models in the library.

To ensure quality of the submitted models a review and rating process is developed. Based on the developer guidelines, a reviewer will be able to check a model, e.g. if the model is thoroughly documented, if the test coverage is good enough, etc. This feature allows the collection of high quality models and encourages users to submit premature models, which is explicitly desired and can be of extra value for other users.

Technically, the database runs an index server, which provides a full text search engine and a web site front-end for the user to search and directly download models from the central repository server. A prior version was based on a decentralized approach using the distributed source control management tool Mercurial [20], where each participating party could manage its own repository server [21]. However, a centralized architecture with a detailed access management system fits better into DLR's current IT infrastructure.

To create or modify models a standalone application is developed which is depicted in Figure 5. It helps the user to manage their own models and update models of other users. It is based on the configuration management system Subversion [22] to keep track of changes. This enables users to collaboratively develop or enhance simulation models prior of their release. Several model developers can work collaboratively on one model by using the features of a source code management system like Subversion.

The chosen database approach seems to be a promising way to promote the reuse of simulation models by supporting developers with best practices and a reliable model development process. In addition, an easy-to-use tool set for collaborative development, management and submission is currently developed. The ordinary users can search and download models simply with a web browser and reuse models at the stage in the process of Figure 3 where it is appropriate because all documentation artifacts are stored alongside the model.

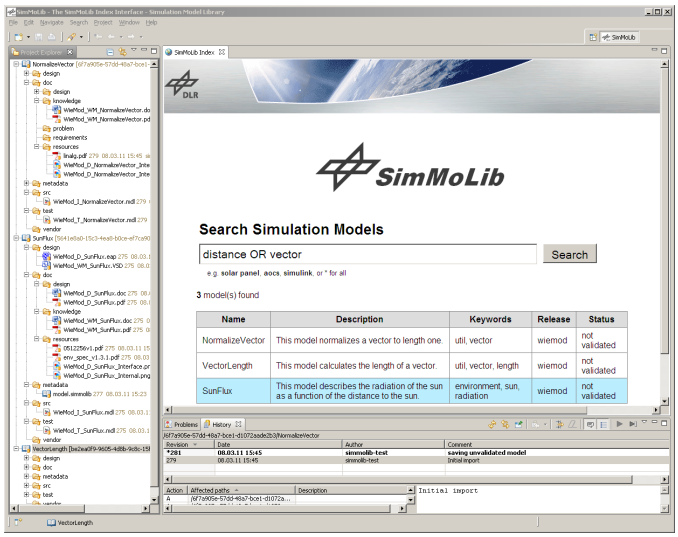

Fig. 5. Screenshot of the model library standalone application with the integrated web view.

V. Conclusion

This paper proposes a sequential, five-step process to develop simulation models. This process requires the creation of different forms of documentation along the way and allows structured collaboration along the development of simulation models. The practical application demonstrates that following this process leads to high quality and well-documented simulation models. Furthermore, it enables partial reuse of simulation models at each step of the process. The lack of integrated tool support throughout the process resulted in time-consuming manual reviews and consistency checks. These issues need to be addressed in the future. Setting up a model database to hold all this data is a challenging task, especially considering space projects conducted by large research organizations and SMEs. The current initiative uses a centralized database approach with search engine and flexible meta data profiles to care for various different file types that need to be stored. A user-friendly, web-based search, a future rating system, and dependency management offer easy access to the knowledge of previously defined models.

Acknowledgment

The research and development presented in this paper is part of the WieMod project (reusable simulation models for the virtual product development) and is funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept “KMU innovativ” (grant number 01IS08015x). The authors would like to thank all project partners involved besides AFW and DLR, notably Industrielle Steuerungstechnik GmbH, Institute for Control Engineering of Machine Tools and Manufacturing Units at University of Stuttgart, and itemis AG.

References

[1] C. Haskins, *Systems Engineering Handbook, Version 3.1*. International Council on Systems Engineering (INCOSE), August 2007, document number: INCOSE-TP-2003-002-03.1.

[2] VDI, “Design methodology for mechatronic systems (VDI 2206),” VDI, Tech. Rep., 2004.

[3] *ECSS-M-ST-10C Space project management - Project planning and implementation*, ESA Std., Rev. 1, March 2009. [Online]. Available: www.ecss.nl

[4] P. Fritzen, S. Kranz, P. van der Plas, and M. Arcioni, “Rapid simulation development for concurrent engineering,” in *SpaceOps Conference*, Rome, Italy, June 2006.

[5] V. Schaus, P. Fischer, D. Lüdtke, A. Braukhane, O. Romberg, and A. Gerndt, “Concurrent engineering software development at the German Aerospace Center - status and outlook -,” in *Proceedings of the 4th International Workshop on System & Concurrent Engineering for Space Applications (SECESA)*, October 2010.

[6] M. Bandecchi, B. Melton, B. Gardini, and F. Ongaro, “The ESA/ESTEC concurrent design facility,” in *Proceedings of the 2nd Concurrent Engineering Conference (EuSEC)*, Munich, Germany, September 2000.

[7] H. Schumann, A. Braukhane, A. Gerndt, J. Grundmann, R. Hempel, B. Kazeminejad, O. Romberg, and M. Sippel, “Overview of the new concurrent engineering facility at DLR,” in *3rd International Workshop on System & Concurrent Engineering for Space Applications (SECESA)*, October 2008.

[8] J. Smith, “Concurrent engineering in the Jet Propulsion Laboratory Project Design Center,” *Aerospace Manufacturing Technology Conference & Exposition*, 1998.

[9] A. Hein, R. P. Lopez, S. Herzig, and M. Brandstätter, “Object-oriented system models in spacecraft design: First steps towards an application in Phase B,” in *Proceedings of the 4th International Workshop on System & Concurrent Engineering for Space Applications (SECESA)*, October 2010.

[10] H. Eisenmann, V. Basso, J. Fuchs, and D. de Wilde, “ESA virtual spacecraft design,” in *Proceedings of the 4th International Workshop on System & Concurrent Engineering for Space Applications (SECESA)*, October 2010.

[11] M. Pidd, “Simulation software and model reuse: A polemic,” in *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[12] K. Brieß, W. Bärwald, H. Kayal, and W. Halle, “The BIRD mission,” in *5th ONERA-DLR Aerospace Symposium ODAS 2003*, Toulouse, France, June 2003.

[13] S. Eckert, S. Ritzmann, S.Roemer, and W. Bärwald, “The TET-1 satellite bus - a high reliability bus for earth observation, scientific and technology verification missions in LEO,” in *Small Satellites Systems and Services - The 4S Symposium*, Madeira, Portugal, June 2010.

[14] OMG, *Model Driven Architecture (MDA) Guide*, 2003, document number: omg/03-06-01.

[15] ——, *OMG Systems Modeling Language (OMG SysML^{TM})*, 2010, document number: formal/2010-06-01.

[16] W. Ley, K. Wittmann, and W. Hallmann, *Handbook of space technology*, ser. Aerospace Series. Wiley, 2009.

[17] K. Großekatthöfer and C. Raschke, “Support of ACS development and test by dynamic simulation models,” in *8th IAA Symposium on Small Satellites for Earth Observation*, Berlin, Germany, April 2011.

[18] M. Hitz, H. Werthner, and T. I. Ören, “Employing databases for large scale reuse of simulation models,” in *WSC ’93: Proceedings of the 25th conference on Winter simulation*. New York, NY, USA: ACM, 1993, pp. 544–551.

[19] Y. Teo and C. Szabo, “CoDES: An integrated approach to composable modeling and simulation,” in *41st Annual Simulation Symposium*. IEEE, 2008, pp. 103–110.

[20] B. O’Sullivan, *Mercurial: The Definitive Guide*. O’Reilly Media, 2009.

[21] C. Kerl, “Realisierung einer verteilten Bibliothek für wiederverwendbare dynamische Simulationsmodelle (in German),” Bachelor Thesis, Duale Hochschule Baden-Württemberg Mannheim, September 2010.

[22] C. Pilato, B. Collins-Sussman, and B. Fitzpatrick, *Version control with Subversion*. O’Reilly Media, 2008.