Model Linking to Improve Visibility and Reusability of Models during Space System Development

Meenakshi Deshmukh
German Aerospace Center (DLR)
Simulation and Software Technology
Lilienthalplatz 7, 38108, Braunschweig, Germany
+49-531-295-2078, meenakshi.deshmukh@dlr.de

René Schwarz
German Aerospace Center (DLR)
Institute of Space Systems
Robert-Hooke-Str. 7, 28359 Bremen, Germany
+49-421-24420-1165, rene.schwarz@dlr.de

Andy Braukhane
German Aerospace Center (DLR)
Institute of Space Systems
Robert-Hooke-Str. 7, 28359 Bremen, Germany
+49-421-24420-1123, andy.braukhane@dlr.de

Rosa París López
German Aerospace Center (DLR)
Institute of Space Systems
Robert-Hooke-Str. 7, 28359 Bremen, Germany
+49-421-24420-1221, rosa.parislopez@dlr.de

Andreas Gerndt
German Aerospace Center (DLR)
Simulation and Software Technology
Lilienthalplatz 7, 38108, Braunschweig, Germany
+49-531-295-2782, andreas.gerndt@dlr.de

***Abstract*— The development of space systems involves complex interdisciplinary systems engineering. To manage such complexity, simulation and calculation models are becoming an integral part of it, where different domain-specific models (power, thermal, structure, propulsion, communication, etc.) are developed using different tools. Every domain model contains valuable knowledge of a respective discipline. However, creating such models takes an ample amount of time and efforts. Therefore, a common management for these models is needed to preserve the knowledge and to reuse them in future space missions. The project *Simulation Model Library* (SimMoLib) at the German Aerospace Center (DLR) develops guidelines and best practices regarding model development, model documentation, validation and verification, as well as model reviews to establish a collection of reusable models. To efficiently catalog the models, an innovative software system is created to support collaborative development, submission, archiving, review, search, and utilization of models. In SimMoLib, a model linking concept has been developed and implemented to enhance the model search and their probable reuse. Along with regular keyword-based search, a direct and an indirect linking between the models in the library has been implemented. Therefore, the model linking increases the visibility and consequently promotes the reuse of single and interdependent models within the library. The paper further describes different types of model relationships, categories, hierarchical levels of model development, implementation and presentation of model linking in detail.**

TABLE OF CONTENTS

978-1-4799-1622-1/14/$31.00

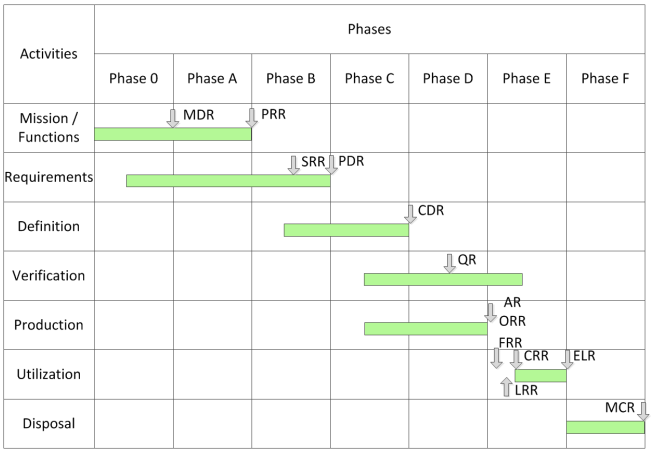


Figure 1. Chart of the space mission development phases according to the ECSS

1. INTRODUCTION TO SPACE MISSION DEVELOPMENT

The development of space systems involves complex interdisciplinary systems engineering. Space mission projects are typically expensive, of long duration and they involve a lot of people from various domains such as power, communication, thermal, payload, structure, propulsion, etc. For the effective management of such big projects, the lifecycle of a space mission has been divided into seven phases as shown in Figure 1 by the European Space Agency (ESA) [4].

While executing space mission projects, the problems such as an interdisciplinary communication, data exchange, interface definitions, cross domain dependency management, verification and validation (V&V) process are very common. For improving the space project management, many space agencies and companies are choosing a concurrent engineering (CE) approach over the traditional sequential engineering [5]. Concurrent engineering enables experts of all involved disciplines to participate in the design process right from the

beginning. The concurrent engineering process has already been applied successfully for pre-phase-A (feasibility) studies [5]. Now the use of this approach in later phases of space missions is also under consideration [6]. The CE approach has reduced the space mission study duration and costs. It has significantly improved the interdisciplinary communication and data exchange.

The information management is still a bottleneck during the space projects. Because of the classical document-centric approach, the information is spread across different documents. Therefore it becomes less comprehensible. It is difficult to maintain consistency as well as traceability and it causes the problems as stated above. To overcome these problems, the model-based systems engineering (MBSE) approach is a promising solution [7]. MBSE provides clear and unambiguous definitions of behavior, capability, and design. It supports a complete representation of a system and traceability through hierarchical system models.

In case of a space mission, a system model is consisting of many subsystems, components, and equipments models. These models are developed using different tools and during different phases of development. To take all benefits of MBSE, a systematic process and management for all these models is required during space mission development. Also each model contains valuable knowledge of a respective domain. Consequently, a platform that manages this knowledge is needed. Such a platform will not only preserve the simulation models and make them accessible and reusable, but also enable the users to individually and collaboratively develop the models throughout the design phases.

This paper gives an overview about the considerations within the Simulation Model Library (SimMoLib), a tool developed at the German Aerospace Center (DLR) for a common management of simulation and calculation models, concerning the linking between simulation models. The paper is organized as follows: Section 2 explains background information about modeling and simulation in space mission development and challenges faced by the domain experts. SimMoLib is briefly introduced in Section 3. In Section 4, a concept for model linking to improve the visibility within SimMoLib and — in turn — the reuse of models during space mission development is described. Two use cases where model linking has helped with the development of complex models are illustrated in Section 5. Finally, the paper ends with a conclusion and insight to future work in Section 6.

2. Modeling and Simulation

To manage complex space projects, simulation and calculation models are becoming an integral part of space mission development. Modeling and simulation is playing an important role in design evaluation. It shortens the design time and reduces the project cost. It also allows for verification and validation of a system before its actual commissioning which results into a high-quality system design.

In the recent years, the use of computer-aided domain-specific modeling has been increased. 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 or thermal engineers setup a simulation based on a geometry model, the orbit, the exposure to direct sunlight, and internal heat sources. Each domain expert uses their domain specific tools for modeling and simulation. For example, MATLAB/Simulink is a standard tool used by control engineers while ESARAD/ESATAN is used by thermal engineers.

Model development takes an ample amount of time and efforts. Moreover, every domain model contains valuable knowledge of a respective domain. Thus, the use of knowledge management techniques can cover the needs of

- gathering the simulation models,
- sharing the expert's know-how that lies within the models,
- making the models reusable, so they can be modified, updated, or adapted to several phases or requirements

throughout a space mission design process [9].

The project called *Wiederverwendbare Modelle (WieMod)* (English: reusable simulation models for the virtual product development) [12] was initiated at the Department of Simulation and Software Technology at DLR to develope a flexible tool-based model development process. A knowledge-based framework for model repositories with unified model descriptions and metamodels was developed. While analyzing the model development in the context of space projects, it was realized that some specific aspects such as multidisciplinary dependencies, phase-to-phase transition, data exchange between multiple teams, etc. need to be considered. A special attention should be given to the collaborative model development within and across different teams. Because of large team sizes, many times the people working on these models are not always aware of other, already existing models. A lot of time is spent in redeveloping the same models. Usually the complexity is managed via incremental refinement of models to improve their fidelity. To maintain traceability and consistency from abstract system models to detailed domain-specific models, a concept for maintaining model relationships is also needed.

Therefore, there is a need to have a single and platform-independent source for model management and development for ongoing space missions and for maintaining a valuable knowledge base for the future ones.

3. Simulation Model Library

To address the need of a common model management for collaborative development, knowledge preservation and model reuse, the Department of Simulation and Software Technology and the Institute of Space Systems at the German Aersospace Center (DLR) have jointly developed a model library called *Simulation Model Library* (SimMoLib) [8]. As a part of SimMoLib, modeling guidelines and best practises have been prepared to ensure the development of high-quality models. This includes a set of common as well as platform-specific (e.g., MATLAB/Simulink, Excel) guidelines for model developers. A documentation template is provided for proper documentation covering all aspects of the model. Such a documentation forms a good basis for future reuse and adaption. Along with model development, the verification and validation of a developed model against the requirements is equally important to a high quality. To achieve this, SimMoLib provides an automated test framework. These tests are submitted to the library along with the model. Therefore, a user can run these tests and gain confidence in the functionalities of a model before reuse.

A client-server-based software framework allowing collabo-

rative model development is developed as a part of SimMoLib in order to collect the models along with their source code, documentation, and tests and to catalog these models for an easy search and reuse. An eclipse-based rich client serves as a classical desktop application. Using the desktop client, a developer can add model files, documentation, and tests along with the metadata to the model project and share this project with other developers to collaboratively improve the model. The intermediate versions of the model under development are managed using a Subversion (SVN) repository [3]. Once the model reaches a certain maturity, it can be released and then it becomes visible to all the other users and it can be downloaded if the user has the required access rights. Released models can be reviewed or further developed. Figure 2 shows the screenshot of the SimMoLib desktop application.

The models under development and the released models are managed by the SimMoLib server. The server provides a web interface for the users who want to search and browse through the model library. They are able to directly download the ZIP file of the required model for further use. Figure 3 shows a screenshot of the SimMoLib web client displaying available models from the library.

SimMoLib covers the whole model lifecycle from the high-quality model development to the storage and cataloging of models. While developing SimMoLib, it has been realized that only providing a model collection in a library is not sufficient. To increase reuse, an efficient search for models is required while browsing through the model database. Therefore, along with standard keyword-based search, a more advanced search, which is able to display suitable models as well as to suggest users related models, is required. To achieve this, a model linking concept has been developed and implemented in SimMoLib to enhance the model modularization, search and their probable reuse.

4. CONCEPT OF MODEL LINKING

As explained in the previous section, model linking was implemented to improve the model search and their reuse within the library. The concept evolved based on some classical online portals like *Amazon*, *booking.com*, *eBay*, etc. Consider the online shopping portal *Amazon*: A user enters some keywords about shopping items in the search window of Amazon. Then the items satisfying the keywords are listed and displayed. When the user chooses a particular product, also related products like

- "customers who bought this item also bought",
- "frequently bought together", or
- "customers who viewed this item also viewed"

are displayed. This provides further assistance to the users to help them finding the product appropriate to their needs.

In case of SimMoLib, a similar idea arose where, along with the keyword-based search, the models related by domains, applications, or by some other relations are also displayed to the user. Because of such an advanced result display, users get acquainted with more and more models from the library. This increases the chance of finding useful models. A model linking concept shall not only help by searching for related models, but also facilitate building upon existing models by creating extensions, add-ons, improving accuracy and functionality. During the concept development and the prototype implementation, two types of linking between the models have been established. One is *direct* linking and the other is *indirect* linking. Both types are explained in detail in the following subsections.

Direct Linking

The establishment of relationships between the models in the library by explicitly adding references to related models is referred as *direct linking*. Within the model linking concept of SimMoLib, one-to-one unidirectional and one-to-one bidirectional links have been found to be appropriate for representing links between the single simulation and calculation models. *One-to-one*, in this context, implies that the logical link will only be established between two atomic models, but not between an atomic model and a group of models (one-to-n link) or between two groups of models (n-to-n link). A link always represents a certain *relationship type*. Several relationship types have been investigated (cf. Table 1) and it was found that there are two classes of relationships: While a relationship between two models A and B might be interpreted from the perspective of A in the same way as from the perspective of B (e. g. if model A is an alternative to model B, model B is also an alternative to model A) and hence the resulting link is a *non-directed* or *bidirectional* link, the other group of relationships must be interpreted differently according to the perspective considered (e. g. if model A depends on model B, then model B is a required condition for model A to be executed, but not necessarily the other way around) and is therefore resulting in *directed* or *unidirectional* links.

Such unidirectional links might be established from newly added or further developed/extended models to already existing models, i. e. in *retrograde* direction, or from already existing models to later added or further developed/extended models, i. e. in *prograde* direction. In SimMoLib, it is only possible to define retrograde unidirectional links explicitly, while all prograde unidirectional links are implicitly inferred by the software. There are two reasons for handling unidirectional linking this way: In the first place the consistency and persistency in the resulting mesh of links needs to be maintained. Additionally, allowing prograde unidirectional links would give rise to ownership issues. In order to explain these two reasons in more detail, one might consider the situation that a developer has submitted model A to the SimMoLib repository. Later on a second developer submits a model B, which depends on model A, to the library. The developer of model A will not know if or when model A has been included into another model, i. e. when model A becomes a required part of another model. Instead, only the developer of model B knows that model B needs this specific model A. So the retrograde unidirectional link "depends on" should be explicitly established by the developer of model B on model B to A, while the reverse prograde unidirectional relation "is part of" from model A to B should be implicitly inferred as reverse relationship by the system automatically, without having an explicit link set (consistency and persistency). Additionally, every developer should only be able to edit the details of his own models in SimMoLib. In this scenario, the developer of model B would have to change the properties of model A to add the prograde unidirectional relationship "is required by" for model B to model A. Since model A is not in his ownership, he should not be able to edit its properties.

The relationships considered in SimMoLib are enlisted in Table 1; relationships I to V are of unidirectional type, while VI and VII are of bidirectional type. The relation "depends on" (I) can be used to show the composition of a complex model using a set of simple models. If a model has been

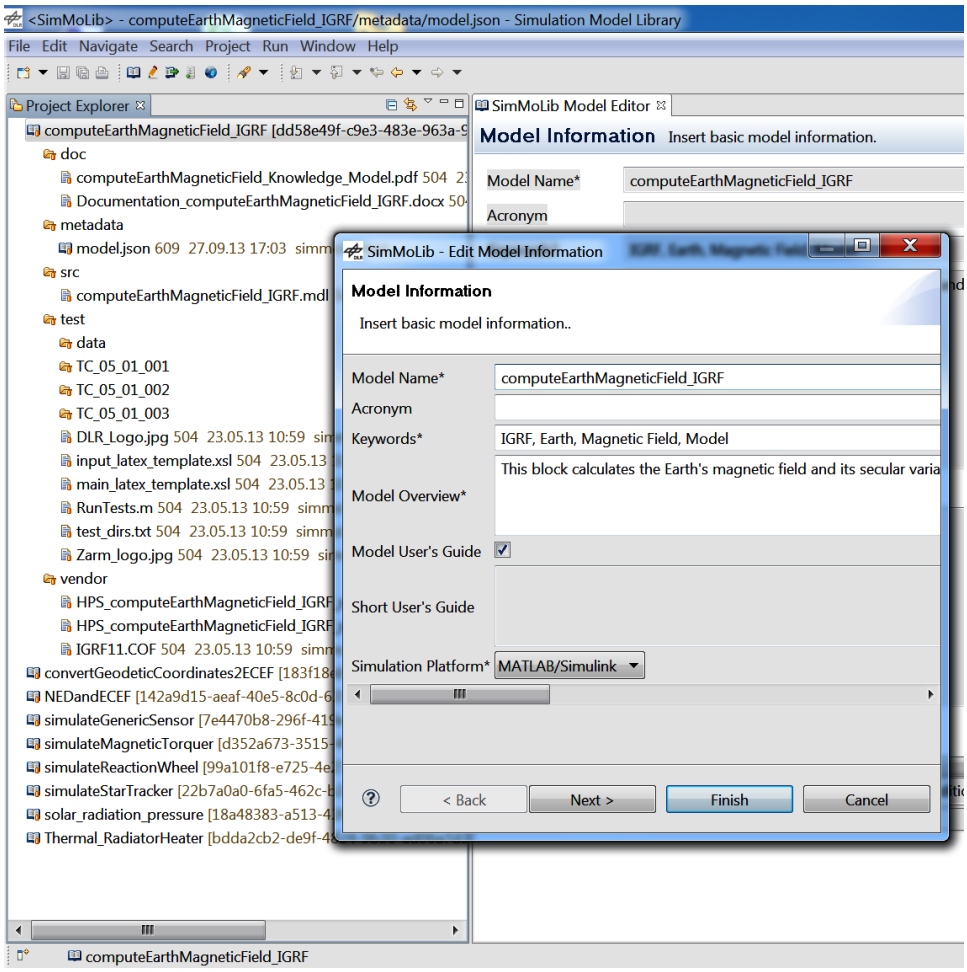


Figure 2. Screenshot of SimMoLib's desktop client application.

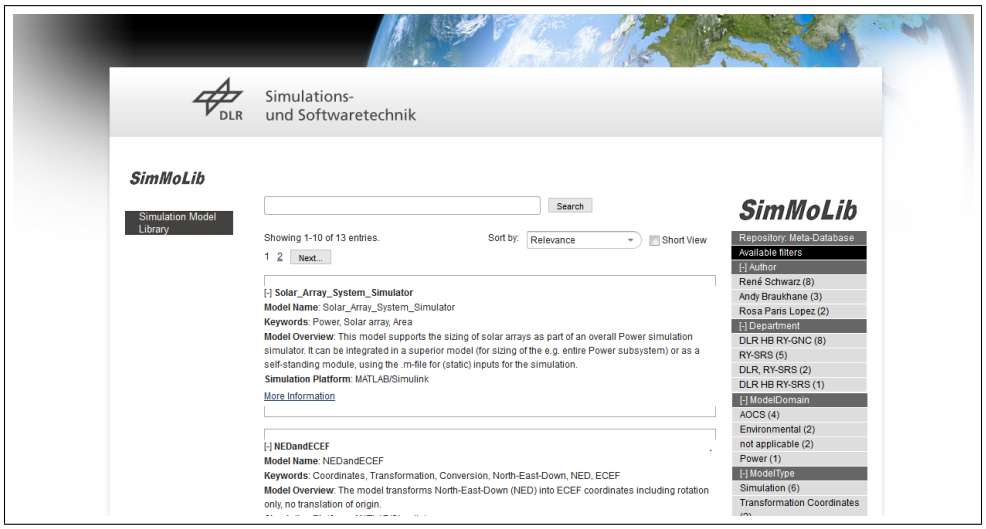


Figure 3. Screenshot of SimMoLib's web interface for model search.

Table 1. Relationship types between the models considered in SimMoLib.

No.	Relation Type	Explanation (Model A ¯ Model B)
I	"depends on" reverse relation: "is required by"	Model B is a required part of model A; model A cannot be executed without model B (necessary condition). **Example:** `Power.mdl ¯ Battery.mdl`
II	"has been derived from" reverse relation: "is a template for"	Model A is based on model B, but there are interface changes or the overall behavior of the model changed (f.e. the model is now used to calculate a different physical scenario). **Example:** `convertMJD2Date.mdl ¯ convertJD2Date.mdl`
III	"is an extension to" reverse relation: "has been extended by"	Model A extends the capabilities of model B. **Example:** `LinkBudget.xls ¯ DataRate.xls`
IV	"is a reimplementation of" reverse relation: "has been reimplemented with"	Model A has been reimplemented for another simulation platform from model B; its function is the same as model B. **Example:** `Orbit.mdl ¯ Orbit.xls`
V	"is advanced to" reverse relation: "is simpler to"	Model A is a refined version of model B to provide a higher level of detail. **Example:** `MultiBodyDynamics.mdl ¯ SimpleDynamics.mdl`
VI	"is an alternative to"	Model A is an alternative model to model B, providing a similar function, e. g. with different algorithm. **Example:** `MagneticFieldIGRF.mdl ¯ MagneticFieldTsyganenko05.mdl`
VII	"is compatible to"	Model A can be used in combination with model B (e. g. because they belong to a common scenario or because the interfaces are the same). **Example:** `Power.xls ¯ SystemSimulator.mdl`

used as template for a model similar in kind, the relation "has been derived from" (II) can be used, while the relation "is an extension to" (IV) describes the extension of this template by additional functionality. Because SimMoLib supports simulation and calculation models for different simulation platforms, a relationship type for models rewritten for other simulation platforms is provided with relationship type "is a reimplementation of" (IV). The relation "is advanced to" (V) describes the model development from abstract to detailed level. The relation "is an alternative" (VI) is used to show variants of a single model, while "is compatible to" (VII) indicated that two models can be used within a common simulation to gain additional functionality/detail.

The direct model linking shall not only help by searching for related models, but also facilitate building upon existing models by creating extensions, add-ons, improving accuracy and functionality.

This theoretical concept of direct linking is implemented into SimMoLib by allowing a developer to manually link a model to other models in the SimMoLib desktop application during model submission to the library or while editing the model properties. Before submitting a model to the library, the developer has to fill out the model's metadata information. Figure 4 shows a screenshot of the SimMoLib desktop application with a dependency information-related metadata form page. Here the developer can select the type of the relation and the related models. This establishes a direct link between the current model and already submitted models. Whenever a user browses through SimMoLib web interface, the related models can be viewed by clicking the "More Information" button on the web page.

Indirect Linking

As described earlier, the developer has to fill out metadata information of the model before submitting it to the SimMoLib repository. The metadata fields include the developer's details, domain, simulation platform, application type, development phase, etc. Based on these metadata fields, an automatic classification of the model is performed and displayed to users using filters. Such kind of indirect linking gives information of possibly related models from the available models. For example, all models from the domain AOCS, all models performing coordinate transformations, or all models from a particular developer, etc. can be displayed.

Figure 5 shows a screenshot of the web interface of SimMoLib with model classification based on four filters, namely "Author", "Department", "Model Domain", and "Model Type". Such suggestions through filters increase the visibility of models submitted to the library. For example, during the concurrent engineering session for a mission feasibility study, a domain expert of the power subsystem can browse through all the available power models by clicking on the "Power" filter and decide whether to use an existing model or to develop a new one. In another case, if an AOCS domain expert wants to browse all AOCS models performing coordinate

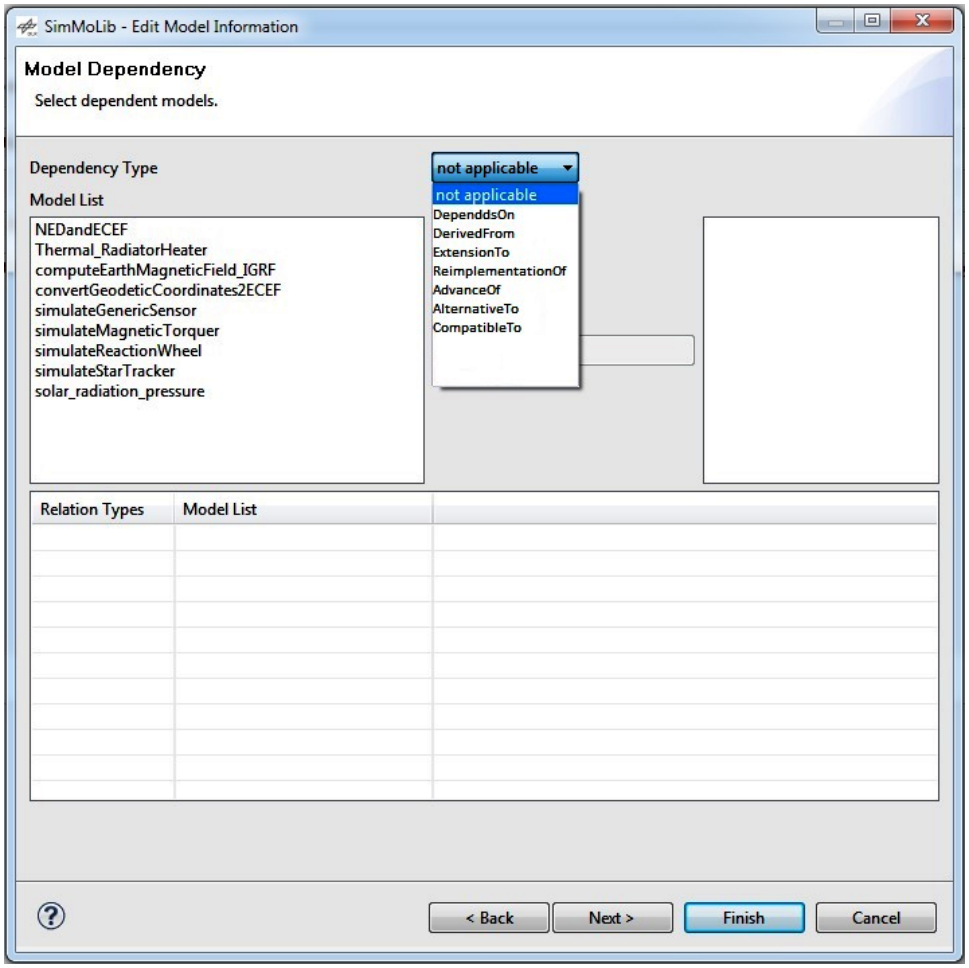


Figure 4. Screenshot of SimMoLib's desktop application with dependency information metadata dialog displayed.

transformation, then one can apply two filters "AOCS" from model domain and "Coordinate Transformation" from model type. This helps in narrowing down the search to the specific requirements of a developer. It provides more alternatives to the user and this in turn increases the probability of model reuse.

5. Application Scenarios

Within this section, two application scenarios showing the utilization of the model linking concept in the context of model development in the space domain are described.

GNC Algorithm Development

In joint cooperation between the Department of Guidance, Navigation and Control Systems at the DLR Institute of Space Systems, Bremen, and the Department of Spaceflight Technology at DLR Space Operations and Astronaut Training, Oberpfaffenhofen, a demonstration model for SimMoLib has been developed. This demonstration model is a MATLAB/Simulink simulation of a formation flight scenario of a target and a chaser satellite in a low Earth orbit with focus on the guidance, navigation, and control (GNC) system for such a scenario (cf. Figure 6 for a screenshot of the Simulink main layer). Although this model actually contains all the basic

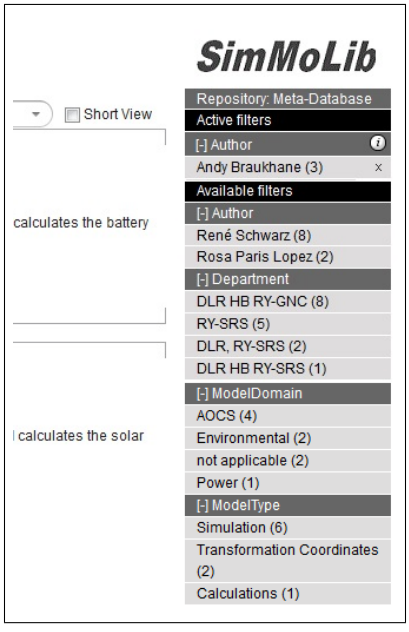


Figure 5. Screenshot of SimMoLib's web interface with filter-based model classification.

elements needed in a real GNC simulation, it was mainly intended as a test for SimMoLib concerning the work with the SimMoLib client, library, and guidelines from a developer's perspective. The experiences gained with the work on this scenario have been retransferred to the SimMoLib concepts for modeling, user interaction, and interfaces as well as for model linking.

Prior to assembling the different simulation models in a superior simulation, the required simulation models had to be created and submitted to the SimMoLib model repository. Most of these models were already existing either in form of simulation models originating from prior missions/simulations or in form of modules within the High Performance Satellite Dynamics Simulator (HPS) [10], jointly developed and maintained by the Center of Applied Space Technology and Microgravity (German: *Zentrum für Angewandte Raumfahrttechnologie und Mikrogravitation*, abbreviated ZARM), Bremen, and the Department of Guidance, Navigation and Control Systems at the DLR Institute of Space Systems, Bremen. All models had to be adapted to the SimMoLib library in terms of file structure, documentation, and metadata information obeying the SimMoLib modeling guidelines. This way, 16 simulation models have been created and committed to the SimMoLib model repository. The models comprise

- simulation models for the orbit and attitude dynamics of both satellites,
- environment models such as the Earth's magnetic field, the drag due to Earth's atmosphere, the gravitational potential as well as the pressure caused by solar radiation,
- sensor models for inertial sensors (gyroscopes, accelerometers) and absolute sensors (magnetometers, GPS receivers),
- actuator models for reaction wheels, magnetic torquers, and thrusters, as well as
- simulation models for GNC algorithms, particularly with regard to formation keeping for the chaser satellite.

Among these simulation models, also calculation and support models for coordinate transformations, angle and unit transformations, simulation initialization, scenario configuration, etc. have been adapted and transferred to the SimMoLib model repository. All these models are closely interconnected and fundamental to compose a complete GNC system simulation. This way, a considerable amount of models only for the GNC domain has to be maintained and model developers have to be aware of interfaces and compatibility to other models. The amount of necessary models explodes when it comes to interdisciplinary simulations, such as a complete system simulation of a satellite or spacecraft, and it is increasingly difficult to keep an overall view.

One may consider a simple example that might illustrate how fast logical relationships between simulation models accumulate and how complex the resulting meshes of direct links get: In the development and simulation of GNC systems, attitude and rotation representations are usually of fundamental importance, e. g. for expressing a spacecraft's attitude or for transformations between different coordinate systems like between a spacecraft-fixed and an inertial coordinate system. There are several representation forms — all having their advantages and disadvantages — in use, e. g. quaternions, direction cosine matrices (DCMs), or Euler angle sequences. There is a need for converting between these representations, because some of them may be more appropriate than others in certain contexts or situations.

In the SimMoLib GNC demo scenario, for example, conversions between quaternions and DCMs as well as between quaternions and Euler angle sequences are required. A calculation model for each direction (quaternion to DCM, DCM to quaternion, quaternion to Euler angle sequence, and Euler angle sequence to quaternion) has been adapted for the SimMoLib repository. Figure 7 shows a graphical overview of the direct links between the models of this example.

Since all calculation models which transform their inputs into quaternions need a quaternion normalization somewhere in the transformation process, this common algorithm has been separated into a new calculation model, thus preventing code duplications, enabling a central code maintenance and, as a consequence, reducing the sources of potential errors. In turn, the quaternion normalization model itself is based on a vector normalization model, as the approach for vector and quaternion normalization is almost similar. For the conversion of DCMs to quaternions, two different algorithms have been implemented: While the algorithm of SHEPPERD [13] is computational less demanding than the one of BAR-ITZHACK [1], it may induce some issues regarding algorithm singularities for certain inputs [13, p. 224]. Nevertheless, both algorithms are in use. Besides that, in the course on the work on the SimMoLib demo scenario, a MATLAB quaternion class has been developed within the HPS. Although this quaternion class has not been adapted for the use in SimMoLib, it has been added to the illustration in Figure 7 to show further logical relationships in the context of quaternion transformation models. The quaternion class contains advanced capabilities for the representation of, calculations with, and transformation between quaternions and other attitude/rotation representations.

The SimMoLib model linking concept helps to gain a quick overview over such logically related models in a common library and therefore facilitates the search, utilization, and further development of models contained within the library. It helps to mitigate the issues arising inevitably with increasing number of simulation models.

Model Modularization

When using models from the very beginning of a project, there is a need for modularization of these models since the level of detail increases significantly throughout the different iterations in a concept and design phase. In space system design, one of the first tasks is to perform a preliminary assessment of the mass budget of the space segment. This requires input from the mission analysis and information about the payload. With this data, a systems engineer can quickly do a first rough assessment of the mass, power demand and envelope of the spacecraft (S/C). A useful tool to support this step is the *Space Mission Analysis and Design* (SMAD) book by WERTZ [14] which provides a baseline for several, often spreadsheet-based, system design models. In this case, a single model covers the information related to the main subsystems (S/S) and links them to provide an initial statement on system level.

For the next design iteration, a higher granularity is required. More detailed questions are answered by domain specialists who do further analysis using more advanced calculation or simulation models. The system design model can be replaced by several subsystem-specific models. However, similar to the first step, the subsystem estimates often start with static and high-level data which is required as input for other disciplines to begin their design tasks. One example is the required solar array area to be defined by the power

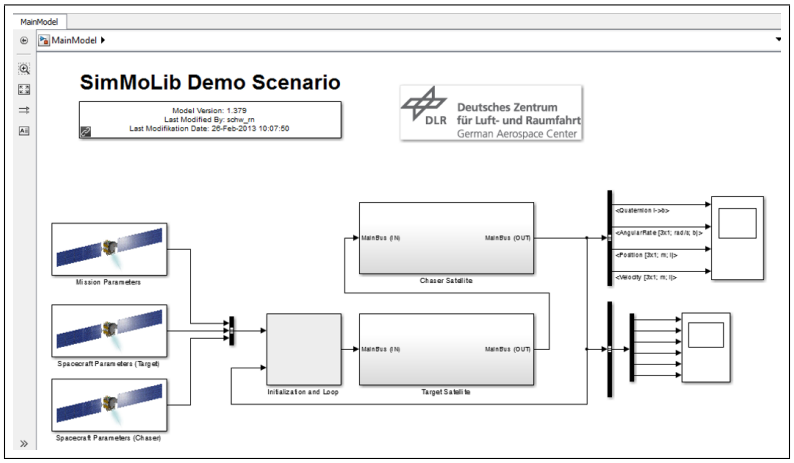


Figure 6. Screenshot of the MATLAB/Simulink GNC demo scenario developed for SimMoLib (main layer).

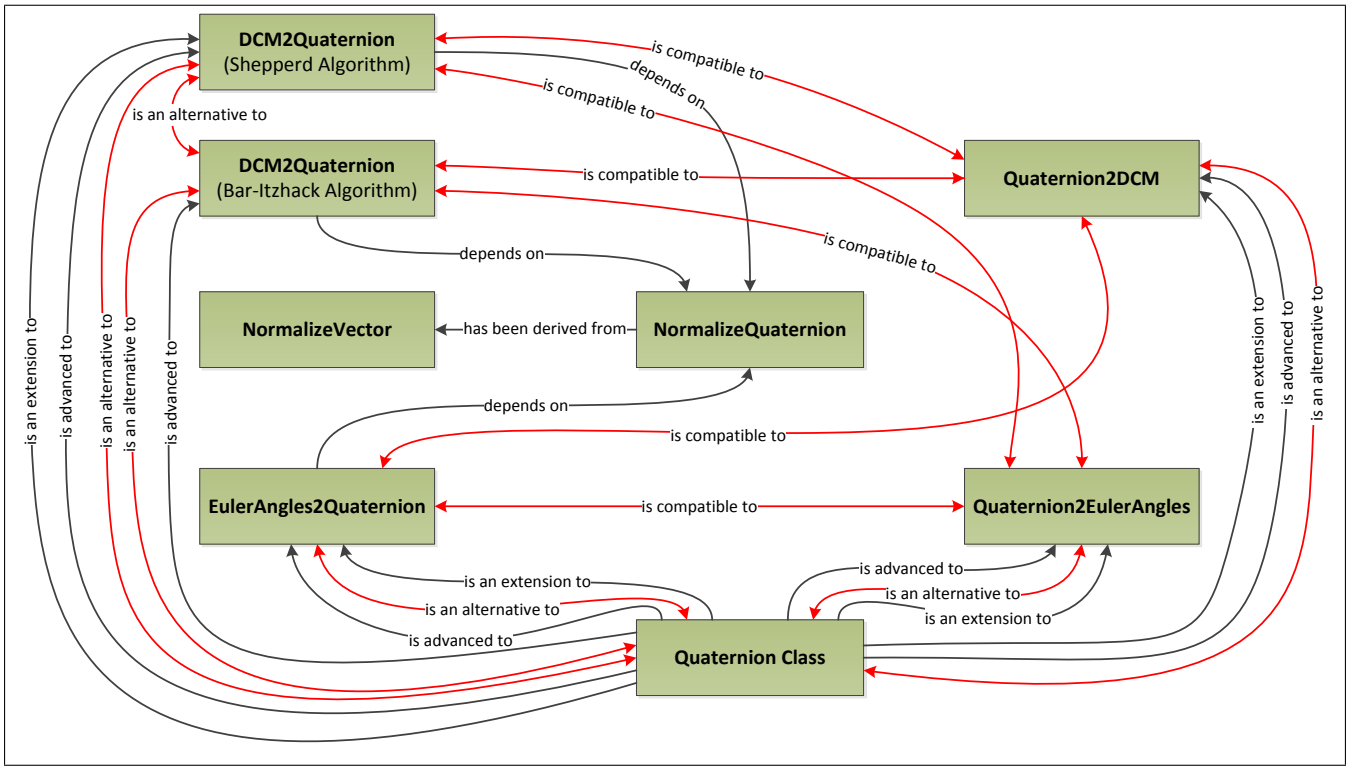


Figure 7. Example for a mesh of direct links between simulation models belonging to a common domain. Red links are bidirectional ones, while black ones are retrograde unidirectional ones. The implicit prograde unidirectional links, i. e. the implied reverse relationships, are not drawn here for the sake of clarity.

engineers which is an input for the structure domain. This can be done with dedicated Excel models covering the design of a specific domain, e. g. a power domain workbook which includes solar array and battery mass calculations.

Within an upcoming iteration, the subject matter experts should provide a more elaborated assessment of their equipment which is to be integrated on the S/C later on. This usually requires a dynamic simulation. In the case of a solar array model, the varying orbital parameters need to be included for a more precise power generation assessment over time. For this step, the S/S calculation model is replaced by one (or more) simulation models.

For example, in the DLR Concurrent Engineering Facility (CEF), a system analysis laboratory for simultaneous and rapid space system design, there is a need of models which address several stages of the development process, including initial system assessment and more detailed S/S design tasks. This requires clear interfaces between subsystems as well as between S/S-components. A central design tool for data exchange called *Virtual Satellite* (VirSat) [11] is used

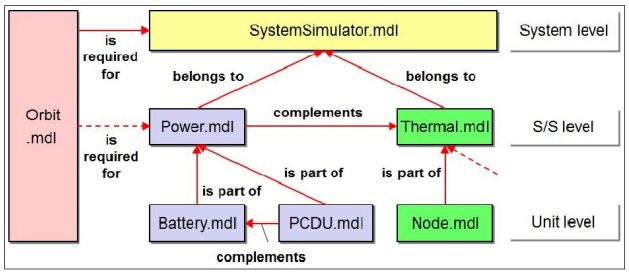


Figure 8. Example of a (MATLAB/Simulink) model hierarchy and their relations

which connects the domains and allows a parallel and easy processing of the data. This tool (or any similar one) can be connected to domain-specific models. In the Virtual Satellite example, a product tree represents the hierarchy of system elements, subsystems, and units, from which each single one can be connected to a dedicated model.

In order to select appropriate models or to build up a superior model customized for a specific space system (or mission), SimMoLib and its model linking feature provides a useful mechanism. It supports the selection and compilation process by addressing and interpreting relations amongst the different models.

Spreadsheet-based models usually do not need direct linking. They can be either used as a self-standing calculation tool or — as done in the CEF — embedded in a data exchange tool where input and output values can be connected manually, as required. For more detailed simulations, different submodels which may replace more high-level models should be equipped with direct linking within the SimMoLib library to ensure that the complementary model set is used. This allows engineers to choose between using (or modifying) only one submodel or a compilation of models. An example is given in Figure 8. Here, models on unit level (e. g. for the battery or thermal node layout) contribute to the S/S design. It is unlikely that these models are running independently, since they have to take into account additional overall inputs (such as orbital parameters) to provide a meaningful output. The subsystem models on the other hand can be equipped with numerous unit models and should work independently, since S/S are often inline with the different specializations of domain engineers. However, the possibility to connect subsystem models on a higher level (here within `SystemSimulator.mdl`) is desired to facilitate an overall picture of the interdependent S/C functions, behavior, and hence the design.

Additional examples for the use cases of model linking and modularization are

- structural models for structural elements to be connected to the parts in e. g. the VirSat tool,
- a system power budget model which is collected on system level but feeds the power and thermal domain models, or
- S/C attitude, orbit, and coordinate transformation models which can be compiled within a mission model to feed the system model (or directly its respective S/S models) with e. g. eclipse phases, view angles, or body orientations.

6. Conclusion and Future Work

The preservation, management and reuse of knowledge is becoming an important aspect during space projects. This paper presents a tool called SimMoLib which focuses on the development and management of simulation and calculation models during a space mission development lifecycle. SimMoLib develops guidelines and best practices regarding model development, model documentation, validation and verification, as well as model reviews to establish a collection of reusable models. To efficiently catalog the models, an innovative software system is created to support collaborative development, submission, archiving, review, search, and utilization of models.

Moreover, a feature like model linking increases the visibility and, consequently, the reuse of single and interdependent models within the library. The model reuse ensures a high quality and a shortened design time, as the submitted models are already documented, tested, and verified. The model linking helps in tracking the model development across different phases of a space mission and at different levels of detail. It provides support for finding people who work on similar topics. It also helps to make simulations/calculations more granular, i. e. it helps to divide big/complex simulations, calculations, or software into smaller modules. Instead of redeveloping the model each time, it is easy to take advantage of already developed models in a systematic way using SimMoLib and its model linking feature.

A review process is an additional feature that was also taken into consideration to enhance model reuse. Using this feature, a user who is using an already submitted model from the library can rate the quality of a model. Some standard rating schemes like star-based or point-based ones have been considered. On the one hand such rating is useful to decide about model reuse, however, on the other hand, it could discourage developers to submit models to the library. Therefore, in future a comment box on the SimMoLib web interface will be added where users can put some comments for the developer in order to refine the model.

In multidisciplinary projects like space missions, every domain uses their domain-specific tools for modeling like MATLAB/Simulink, CATIA, etc. In future, a more user-friendly and direct interfaces between SimMoLib and these tools will be implemented.

Virtual Satellite has become the standard tool for data exchange during feasibility studies conducted at the Concurrent Engineering Facility (CEF) at DLR Bremen [2]. The integration of SimMoLib into Virtual Satellite is in progress. This will enable the linking of models in SimMoLib to the centralized data model of Virtual Satellite and cross-domain dependencies could be efficiently handled.

Acknowledgments

The authors thank Jean-Sébastien Ardaens from the Department of Spaceflight Technology at DLR Space Operations and Astronaut Training, Oberpfaffenhofen, for his participation in developing the SimMoLib GNC demo scenario. We thank the Center of Applied Space Technology and Microgravity (ZARM), Bremen, and the Department of Guidance, Navigation and Control Systems at the DLR Institute of Space Systems, Bremen, for allowing the use of models from the High Performance Satellite Dynamics Simulator (HPS) in SimMoLib. We also thank Daniel Lüdtke from the

Department of Simulation and Software Technology at German Aerospace Center (DLR), Braunschweig, for his active participation during the model linking concept development.

The SimMoLib project is funded by the Program Directorate Space Research and Technology of the German Aerospace Center (DLR).

REFERENCES

[1] Bar-Itzhack, Itzhack Y.: *New Method for Extracting the Quaternion from a Rotation Matrix*. In: *Journal of Guidance, Control, and Dynamics* **23** (6): 1085–1087. American Institute of Aeronautics and Astronautics (AIAA), 2000. ISSN 0731-5090. DOI 10.2514/2.4654.

[2] Braukhane, A.; Maiwald, V.; Quantius, D.; Romberg, O.: *Statistics and Evaluation of 30+ Concurrent Engineering Studies at DLR*. 5th International Workshop on Systems Engineering for Space Applications (SECESA), Lisbon, October 17–19, 2012.

[3] Collins-Sussman, B.; Fitzpatrick, B-W.; Pilato, C-M.: *Version Control with Subversion for Subversion 1.7*. 2011. Online available at http://svnbook.red-bean.com/en/1.7/svn-book.pdf. Accessed January 17, 2012.

[4] ECSS (ed.): *Space project management — Project planning and implementation*. ECSS Standard ECSS-M-ST-10C. European Cooperation for Space Standardization (ECSS), 2009. Online available at http://www.ecss.nl/.

[5] ESA (ed.): *The ESA Concurrent Design Facility: Concurrent Engineering applied to space mission assessments*. CDF Info Pack. European Space Agency (ESA), 2011. Online available at http://esamultimedia.esa.int/docs/cdf/CDF-INFOPACK-2011.pdf. Accessed October 18, 2013.

[6] Findlay, R.; Braukhane, A.; Schubert, D.; Pedersen, J-F.; Mueller, H.; Essmann, O.: *Implementation of concurrent engineering to Phase B space system design*. In: *CEAS Space Journal* **2** (1–4): 51–58. Springer, Vienna, December 2011. ISSN 1868-2502. DOI 10.1007/s12567-011-0013-y.

[7] Jansma, P.A.T.; Jones, R.M.: *Advancing the practice of systems engineering at JPL*. IEEE Aerospace Conference, March 2006, Big Sky, Montana, USA.

[8] Lüdtke, D.; Ardaens, J-S.; Deshmukh, M.; París López, R.; Braukhane, A.; Pelivan, I.; Theil, S.; Gerndt, A.: *Collaborative Development and Cataloging of Simulation and Calculation Models for Space Systems*. IEEE Computer Society Press. 21th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), June 25–27, 2012, Toulouse, France.

[9] París López, R.; Soragavi, G.; Deshmukh, M.; Lüdtke, D.: *Knowledge Management Tools Integration within DLR's Concurrent Engineering Facility*. IEEE Aerospace Conference, March 2–9, 2013, Big Sky, Montana, USA.

[10] Pelivan, I.; Heidecker, A.; Theil, S.: *High Performance Satellite Dynamics and Control Simulation for Multi-Purpose Application*. Presented at the 22nd International Symposium on Space Flight Dynamics, So Jos dos Campos, Brazil, February, 2011. In: *Journal of Aerospace Engineering, Sciences and Applications* **IV** (3): 119–130. Brazilian Aerospace Association (AAB), 2012. ISSN 2236-557X. DOI 10.7446/jaesa.0403.11. Online available at http://173.192.111.37/~aeroe452/jaesa/editions/repository/v04/n03/11-PelivanHeideckerTheil.pdf.

[11] Schaus, V.; Fischer, P-M.; Lüdtke, D.; Braukhane, A.; Romberg, O.; Gerndt, A.: *Concurrent Engineering Software Development at German Aerospace Center — Status and Outlook*. 4th International Workshop on System & Concurrent Engineering for Space Applications, 2010.

[12] Schaus, V.; Grossekatthofer, K.; Lüdtke, D.; Gerndt, A.: *Collaborative development of a space system simulation model*. In: *Proc. 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2011)*: 164–169. 2011.

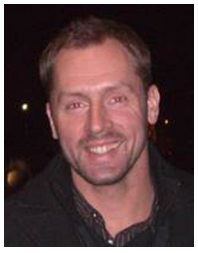
[13] Shepperd, Stanley W.: *Quaternion from Rotation Matrix*. In: *Journal of Guidance and Control* **1** (3): 223–224. American Institute of Aeronautics and Astronautics (AIAA), Mai–June 1978.

[14] Wertz, J. (ed.); Larson, W. (ed.): *Space Mission Analysis and Design*. Space Technology Series **8**. Third edition, tenth printing, 2008. Microcosm Press & Springer, New York, 1999. ISBN 9780792359012.

BIOGRAPHY

Meenakshi Deshmukh *works since June 2011 for the German Aerospace Center (DLR) within the Department of Simulation and Software Technology. She joined the Software for Space Systems section. Since then she is working on the software development projects for supporting the Early Design and Analysis of Space Missions. Her scientific research focuses on the model-based system and software engineering. She has completed her bachelor studies in Information Technology from Pune, India and then pursued Master's degree in Computer engineering from University of Duisburg-Essen, Germany.*

René Schwarz *is a research scientist and engineer at the Department of Guidance, Navigation and Control Systems at the Institute of Space Systems of the German Aerospace Center (DLR). He received his master's degree in Computer Science and Communication Systems/Artificial Intelligence from the Merseburg University of Applied Sciences, Germany, in 2012, as well as his bachelor's degree in Mechatronics, Industrial and Physics Technology in 2009.*



Andy Braukhane *started working for the German Aerospace Center (DLR) in October 2007. He is heading the Concurrent Engineering Facility (CEF) at DLR Bremen as a member of the Space Segment System Analysis department. He acts as a team leader and systems engineer for feasibility studies regarding satellite design, exploration mission architectures or launcher configuration within the CEF. He participated also in studies with non-space-related projects such as aeronautics-, deep water*

system or parking house design He was leading more than 20 space system studies so far. In 2007 he received his diploma in Aerospace Engineering from the University of Applied Sciences in Bremen, Germany and in 2009 a Masters degree of Space Systems Engineering from the Technical University of Delft, The Netherlands. His major background is Systems and Mechanical Engineering.

Rosa París López *works since June 2011 for the German Aerospace Center (DLR) within the Department of System Analysis Space Segment (SARA). She joined the Concurrent Engineering and System Concepts (CESY) section. Since then she has participated in thirteen Concurrent Engineering studies with organization, moderation, systems engineering and specific domain tasks. Her scientific research focuses on the development of simulation and calculation models, as well as the optimization of Concurrent Engineering methods. She received her Master's degree in Aeronautical Engineering with specialization in Space Engineering at the Polytechnic University of Catalonia (Spain) after a 1-year exchange at the Technical University of Munich (Germany), where she worked on the implementation of Model-Based Systems Engineering (MBSE) into the Concurrent Engineering methodology.*

Andreas Gerndt *is the head of the Department Software for Space Systems and Interactive Visualization at the German Aerospace Center (DLR). He received his degree in computer science from Technical University, Darmstadt, Germany in 1993. In the position of a research scientist, he also worked at the Fraunhofer Institute for Computer Graphics (IGD) in Germany. Thereafter, he was a software engineer for different companies with focus on Software Engineering and Computer Graphics. In 1999 he continued his studies in Virtual Reality and Scientific Visualization at RWTH Aachen University, Germany, where he received his doctoral degree in computer science. After two years of interdisciplinary research activities as a post doctoral fellow at the University of Louisiana, Lafayette, USA, he returned to Germany in 2008.*