From Knowledge Transfer to Domain Specific Modeling: A Use Case Driven Simulation Approach from Industry Perspective

Mustafa Aydın[(1)]

[(1)]***Turkish Aerospace Industries, Inc.***
Fethiye Mahallesi Havacilik Bulvari
No:17 06980 Kazan Ankara Turkey
Email: muaydin@tai.com.tr

ABSTRACT

In accordance with her primary role in the national satellite programs of Turkey, TAI (Turkish Aerospace Industries, Inc.) has recently involved in studies to develop satellite integration, verification and validation (IV &V) capabilities by establishing a research facility. With the purpose of reducing time, effort and cost of design, development and deployment of new hardware and software while sustaining reliability, TAI has been evaluating usage of virtual satellites, i.e. simulated systems within the mentioned facility. Already manufacturer of space-qualified equipments like control moment gyroscopes, TAI also aims HIL testing capability as well as obtaining tools for supporting ground operations using the satellite simulators.

This paper summarizes TAI's effort to bring together several standards and industry practices while planning the roadmap of simulating spacecraft systems. The main focus of the paper is how productivity and reuse in multidisciplinary domain simulators can be established, applying a use case driven design approach.

Optimization is required to minimize the efforts of the knowledge embedded in domains, to a common platform where simulated and real systems can interact in a controllable environment. TAI has chosen a path focusing on the development of a graphical modeling tool, to transform efforts of knowledge translation between specific domains and software engineering, to encapsulation of the relevant domain data by domain specific tools by the domain experts. The tool is also considered essential to enable composition of several simulation models to improve reuse and interoperability.

The paper also includes the reference architecture for a proposed spacecraft simulation software suite, with the apparent applications of emerging software engineering concepts and technologies.

EXPERIENCE WITH TRADITIONAL ESA TOOLS AND METHODOLOGY

Introduction

The essence of the simulators highly depends on either the unique functionality simulators can support which is not possible by any other technique or the reduced amount of time and cost while sustaining the same performance compared to other techniques. Therefore, simulators are vital components to support the product life cycle for satellite systems in various phases affecting several roles. The applicability of simulated systems to the specific phase depends on the easy customization according to the use cases of the current phase, interoperating with existing software/hardware. Other than applications throughout phases of a process, another major concern is reusing the system components while implementing simulators for new missions.

Industry Approach

Modeling and simulation (M&S) solutions have versatile potential application areas in the business process of TAI, since TAI has undertaken a role as the integrator of satellite platforms, manufacturer of satellite hardware and software components. To acquire the necessary capabilities to support its engineering process with M&S tools, TAI has initiated an internal research project to evaluate the usage with the ultimate goal of applying simulated products to the other phases. While initial focus was on operational simulators, the area of interest has expanded to usage of simulators in all the product life cycle.

The initial project has been planned with the purpose of modeling AODCS subsystem of a reference satellite which TAI manufactures, to test the functional control algorithms in closed loop. Since the overall simulator requires extensive effort, the objective was to build common components to reuse in the further applications. Hence the project has been implemented on SIMSAT NT and generic models for dynamics and environment were reused from the previous ESA missions, with partnership of a proven ESA subcontractor for simulator solutions. TAI has experienced the traditional ESA development for operational simulators approach, applied in several missions. The approach utilizes the SMI API, which enables software components to be loaded to a simulation environment over the COM interface. The process requires an experienced team of engineers from multiple disciplines and enables rapid development of simulators from scratch, with minimum amount of redundant code.

However some major fallbacks were observed in the development process which will be addressed in the following sections. While the SMP was designed as an infrastructure to enable the binary interoperability between models as appears in Fig. 1, even the generic models do not utilize the SMI interface to communicate with each other, which makes SMI merely an interface to publish the simulation content to the scripting environment and the MMI [1]. The SMI interface also, lacks “unpublish” methods, which also makes dynamic configuration impossible. Absence of standard interfaces leads to major modifications in source code, since the model connections are not loosely coupled. One of its tightest bottlenecks is that, it overrides the OOP paradigm of encapsulation by accessing the simulated data through memory references.

Upon the problems listed above, TAI has developed a lightweight simulation framework with generic interfaces and implemented the interfaces using SIMSAT NT and SMI compliant components. However, when the initial objectives are considered, it will be necessary to migrate to a more mature development framework, possibly by adopting the ongoing progress of SMP2 and REFA promoted by ESA.

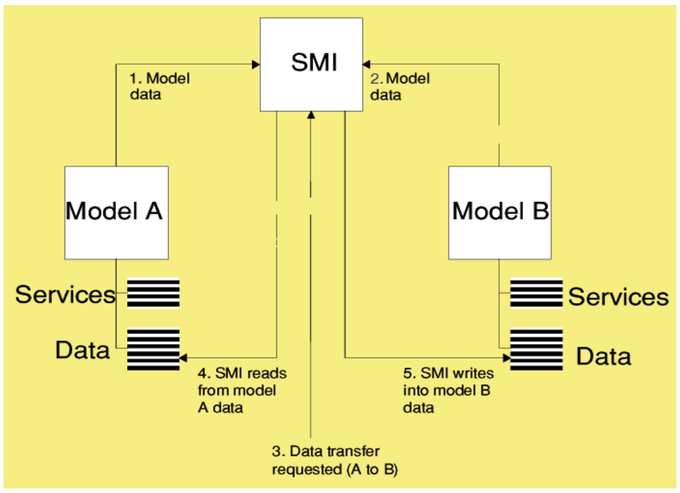
SMI
1. Model data
2. Model data
Model A
Model B
Services
Data
Services
Data
4. SMI reads from model A data
5. SMI writes into model B data
3. Data transfer requested (A to B)

Fig. 1. Desired data transfer using SMP (from ESA Bulletin)

User-centered Design of Simulators

The problems faced in the similar projects are mainly reasoning from the amount of unorganized data to be translated between several engineering domains and software engineering, and the necessity of unique technical capabilities and expertise required to keep this process organized. Engineering simulators are data-intensive applications: They require the data and functionality residing in the simulated domain to be transferred to the application domain through translation of domain knowledge. In the absence of automated tools to perform this task, modeling process requires knowledge transformation in the individual level, either through documentations or verbal methods which is a form of knowledge transfer discussed in [2]. This process is limited by the technology constraints, platform requirements and different software development approaches. The conceptual modeling of the domain may be performed by using plain UML as discussed in [3]. Reference [4] discusses three different representation methods that can be used in modeling.

Discussions over the knowledge transfer and the methodology could be handled concerning different interests. However, most significant interest, the user concern of the engineering simulators is often misconceived, since the development of the tools requires engineering effort as the development resource as well. As a result, the possible utilities required for the development process itself might be considered as user requirements. The consequences of such situation, like conflicting project roles and stakeholders, are alleviated for the teams which are formed in the same organization, i.e. when simulators are built in-site. While in the classical M&S approach experts of domain are hired to access domain knowledge required, in engineering simulators development, software engineers are integrated to the development teams [5].

User centered design of engineering simulators requires intelligent tools, to remove or minimize the dependency and exposure of the simulated domain's experts to the details and constraints of software engineering domain.

- In the modeling phase, traditional methods use re-implementation of the code from the model or equations. Modern methods utilize auto-generation of code from the existing models, interoperation of models implemented on CAE tools or graphical modeling using the graphical tools in the component level modeling. It is obvious that of the modern approaches, the first two require less knowledge transfer compared to the last one.
- In the integration phase, a graphical modeling tool will be discussed upcoming sections, to graphically compose and represent the relations between models. While traditional approach requires integration of the components in the development with glue code, modern approach uses assemblies to represent a set of models to run together. In this phase also easy modification of the simulation configuration to the specified phase requirements, and configuration management of branches resulting from these configuration modifications also should be addressed.
- Once the simulator is constructed, user experience with the simulated data should be considered for the design of modules including scenario/test creation, execution, monitoring, control, recording, debriefing and analysis. Since all the modules stated are related to runtime behavior of the simulator, they are considered out of the scope of this study.

EFFORTS TO IMPROVE PRODUCTIVITY AND REUSE

The Reference Architecture

When the efforts for standardization are initiated from the user perspective described in the previous section, it is obvious that the reference architecture should be the main design driver for the standardization process. The reference architecture, should define the simulation context and decomposition levels, specifications of common models and the basic guidelines of how models will interact with these standard models, not the details of the modeling process itself. Fig. 2 shows, the standard processes with the desired outputs of the simulator building process.

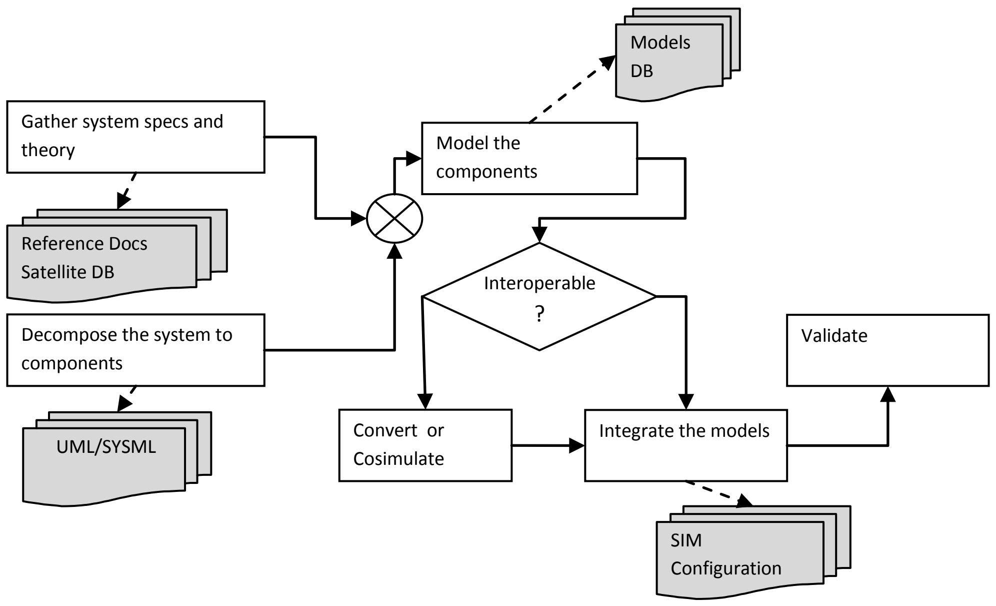

Fig.2. Simulator development process

The reference architecture is specifically required in two processes: decomposition of the complex systems into subsystems and components, and integration of the implemented components to simulate the complete system. The decomposition involves initially listing the entities (e.g. a reaction wheel) that take part in the simulation and defining the common functionality they share, which can be called as the aspects of the entities, like thermal or power aspects. This stage of decomposition process mainly involves identifying the subsumption (is-a) and composition (has-a) relationships as explained in [6]. As the entities and aspects are determined, the designer needs also to define the relationships in the language set of the domain as examples given in Table 1.

The most significant expectation from the reference architecture is that it should standardize the decomposition process and allow each decomposed component or subsystem, to be connected to the overall simulator using the predefined relationships. These standard connections can also be applied to import the engineering data, for example TM/TC database could be imported to simulated system as mentioned in [7]. The lack of such domain specific links in the modeling environment, leads to loss of the engineering data (mostly concerned with systems engineering) in the further process of integration and migration of the components to another mission, since the user cannot edit the relationships without the knowledge of software engineering or the help of a developer once the data is translated.

The elements of a sample reference architecture based on the ETM-10-21 guidelines are presented in [8]. The generation of complete architecture requires collaboration and effort from both different disciplines and industries. While an equipment from the AODCS subsystem and a scientific payload may be isolated in the design and development phases of the actual system until the integration phase, from the M&S perspective they have very similar characteristics. They have effect on dynamics, draw power, affect from environmental and thermal conditions and may be connected through a common bus, and the same building blocks should be used for the implementations of the models representing these equipments.

During the design sessions for the layout of the REFA, interdependency between all aspects shall be carefully examined to avoid possible conflicts. The aspects may have direct effects on each other, like power consumption of a standard communication system. The issue indicates that cross-references are necessary between the aspects. Also the extension mechanisms for custom relationships, will add the capability to interface to the specialized satellite components and custom payloads using the proposed methodology. While a generic model template may be defined with all the available connections, compact models may also be defined using only the required aspects. Despite that the latter approach will result in smaller and more efficient models, the former approach presents a more integrated and easy-to-use model to especially to represent the components which have access to multiple models, as displayed in Fig. 3.

Usage of standard relations/entities in the reference architecture will briefly:

- Standard computation/solver engines: Usage of common solver engines will be promoted, which requires specialization in the related domain.
- Standard models: Development of new models will be standardized by component interfaces. The reuse of the components will be promoted.
- Standard simulation configuration: Simulator configuration will be constructed upon domain level links, which will be exportable with the references to the common components.
- Custom model abstraction levels: Using this modeling strategy, one can replace a very detailed model with complex connections, to only a messaging component through TM/TC connection to OBSW.

Table 1. Sample relations concerning the standard models

Aspect	Relation	Data	Unit
THERMAL SYS	Transfers Heat	Heat	Joule
DYNAMICS	Applies Force To	Force	Newton
ORBIT (PROPAGATOR)	Get Distance To Sun	Length	Meter
FLUID MECHANICS	Transmits Pressure	Pressure	Pascal
ENVIRONMENT	Get Magnetic Field	Magnetic Field	Tesla
ELECTRICAL-POWER	Draw Power From	Power	Watt
RF (TM/TC)	Sends Message To	Telemetry	-
ELECTRICAL-COMM	Connected Through	Bus	-
OBC & EMULATORS	Executes	Instruction	-

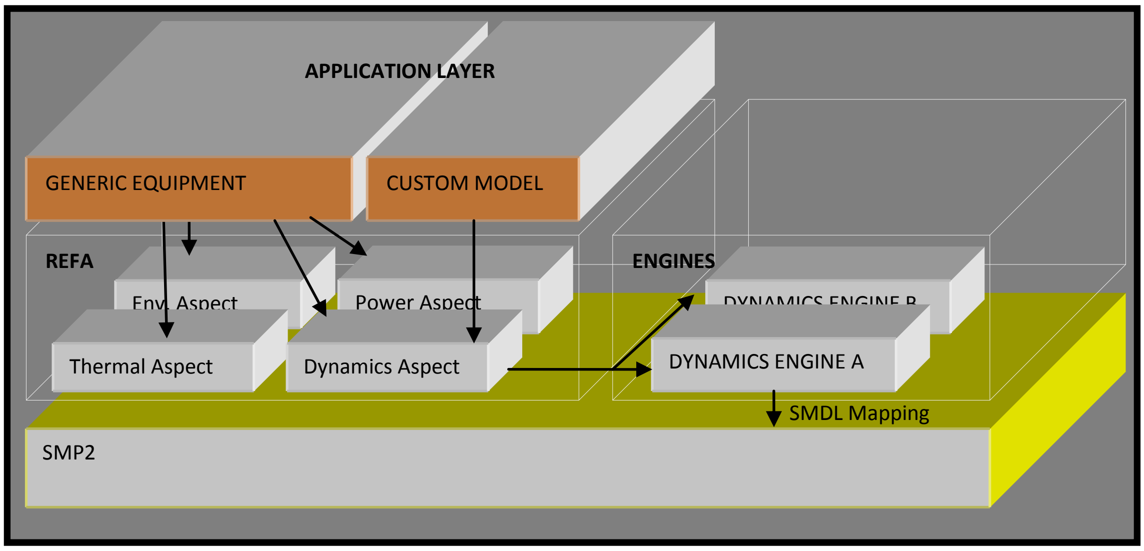

Fig. 3. Layers of the simulated system with focus on reference architecture

Perception of SMP2

The implementation of reference architecture presented above needs a software infrastructure, which applies bridge pattern to glue the generic components to be specialized by the REFA. SMP2 standard defines a meta-model for building platform independent models, which enables low level implementations of simulation models as standard entries in the specified catalogue and assemblies, which can be associated with packages and schedule files. The standard also offers the component model which includes runtime features like common type system, services and event mechanisms. Finally, the standard needs to be supported by utilities like editors, validators to support data exchange in XML format.

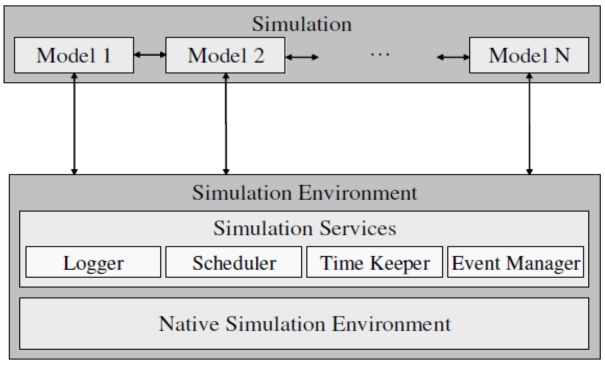

Fig. 4. Typical SMP2 architecture (from SMP2 Handbook)

The standard therefore both has influence on structure and behavior of the models. The standard has been in development since 2004, and the industrial applicability has been subject to experimental validation studies [9]. When compared to SMI, SMP2 stems from modern and mature methodologies from software engineering perspective. However SMP2 does not effectively offer unique functionality for the simulation of the satellite systems. The architecture of typical application implementing SMP2 is given in Fig. 4, which displays the infrastructure and simulated models in separate layers [10]. Although SMP2 enables generic interfaces between the simulated models, inter-model communication mechanisms in SMP2 does not support domain level links. Instead SMDL, an XML based generic modeling language has been adopted. While it is possible to derive and extend the language, it is also possible to make transformations between the standard DEVS models and SMP2 using QVT as mentioned in [11].

HLA may also be considered as an alternative to the SMP2 to implement the reference architecture with an appropriate RTI implementation. Despite the fact that HLA provides sufficient technical capabilities to match the required layer, the design principles of HLA makes it unacceptably complex for such low level interoperability issues. However, mapping of SMP2 to HLA will be a very essential step to interoperate with external simulators not implemented using SMP2. Also SMP2 adapters to common engineering tools like Matlab Simulink, NI LabVIEW are also desired for extensions to integrate the functionality and assets existing in these tools.

The Proposed Composition Tool

Graphical modeling has been mentioned as a technique to improve productivity due to its friendly nature to the end user and applied in several space related M&S activities [12] [13]. The proposed modeling tool is not a comprehensive tool in order to model the system dynamics; indeed it is merely a graphical representation of the REFA previously discussed, with the objective of creating/modifying model compositions. The tool might be designed without being dependent to the details of the underlying architecture; it is even more desirable for a tool to support more than one simulation architecture.

The current trends involve using technologies around Eclipse GMF, which is used to create graphical editors to represent domain specific languages (DSLs). Using DSL's, one can create a platform independent model, which enables generation of template code from the abstract models, which is typical application model-driven architecture (MDA). However the proposed tool is just a graphics editor with plug-in mechanisms to load binary components and link them to create a sophisticated architecture using strategy pattern. As an example the FxEngine, a C++ framework distributed with LGPL license presents a very simple and efficient method to achieve data flow processing and dynamic system design using plug-ins [14]. This framework has two APIs, first for the generation of the plug-ins and second to use them in a host application. A graphical application is also included in the framework, FxEngine Editor, to assemble the implemented plug-ins. Unlike the MDA approach, it is possible to link the executable modules in this editor as plug-ins and run them synchronously. Direct integration of running components also sustains a better environment to debug the system for errors, compared to the possible difficulties which might be encountered during integration of abstract models in UML and code generation/merging as described in [15].

The proposed tool enables to link the models using the standard relations defined in the REFA. Furthermore, showing the relations in the domain of interest through filtered layers is a very necessary feature. This concept is also mentioned for different categories of layers in [16]. In Fig. 5 a screenshot of dynamics aspect in schematic view is presented. Each aspect in the REFA can be grouped in views, to simplify the development, and focus on the related aspect of the simulated system. If the dynamics behavior of the satellite is of interest, then only the components related to dynamics aspect will be displayed. Also this categorization of views might be diversified using different representation methods for each view, like schematic views, tree widgets or simple 3D graphics when possible.

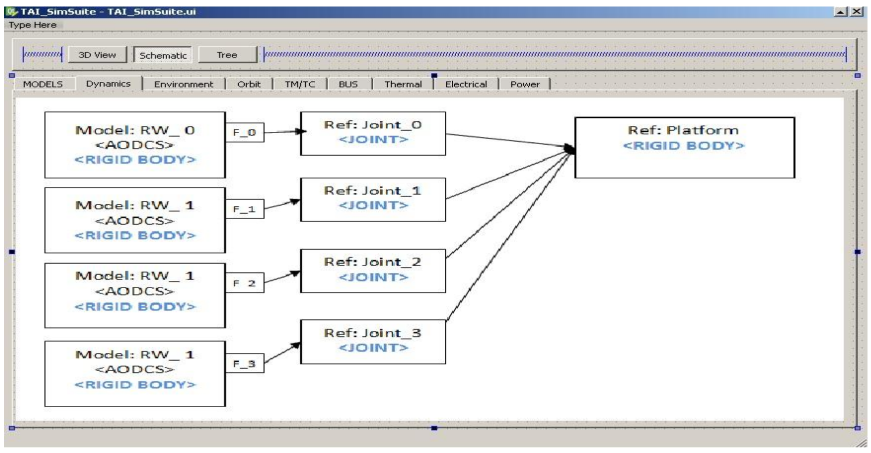

Fig. 5. Dynamics aspect in schematic view

Briefly the proposed tool will have following outcomes:

- Plug-in mechanism to load compatible models without the need to use relatively complex tools.
- Easy integration of the existing models, troubleshooting in domain specific layer from the executable components.
- Refined displays of the simulator architecture, filtered for specific domains.

Analysis of REFA for Dynamics

The objective of the dynamics modeling environment is to standardize the composition of the simulator models that have dynamics aspect, i.e. relationship with the dynamics of the spacecraft. This relationships may be either retrieving data from dynamics, registering, receiving or activating events, and updating data to the dynamics, which are in the component level as described in [17]. In the terms of dynamics domain, these relations might be measuring angular velocity of the platform, applying force or torque to the specific body, changing moment of inertia of a body, colliding with a geometry and contact points, which are independent of any specific implementation.

It is possible to use a single multi-body dynamics engine to compute the dynamics variables in various topologies with different configurations as appears in [18]. However there are variety of alternatives in different formats, and some of these alternatives need to be considered to gather insight into the domain from different perspectives.

- The generic models used by ESA include a library called SIMDYN, which is yet another multibody dynamics engine implemented in Fortran. This library supports only tree topology, and has the heritage of several successful ESA missions.
- Another tool to consider is SimMechanics, an add-on to the de-facto industry standard numerical computing environment Matlab. SimMechanics supports graphical modeling of dynamics systems through definitions of the using block diagrams, instead of formulizing the equations of motions which is common approach in Simulink. This technology is strictly limited to Matlab environment and the only way to utilize this library is to perform co-simulation with Matlab.
- There are also various numbers of physics engines available as free, open-source projects like Ode, Bullet, and Newton as well as proprietary ones like Physx. These engines have different file formats, accuracy, computational efficiency, joints types, constraints and algorithms used for collision detection and contact force generation. Maybe the most impressive tool to mention is Physics Abstraction Layer (PAL) which is an open source, cross platform effort to provide a unified interface to the most popular physics engines [19]. There exists even a graphics editor with a user-friendly interface called Scythe, capable of modeling the physics behavior of a system, which can generate file formats compatible with several physics engines [20].

The REFA could synthesize a common representation for the dynamics aspect of the simulated components, with refinement of common entities and relations, which can easily be abstracted from other aspects of the simulated components. Implementations using the candidates above as test cases could be used for the validation of dynamics aspect of the REFA.

CONCLUSION

Due to limited number of missions and their critical nature, simulators used in space industry may not need the flexibility and ease-of-use as found in the current state-of-art game engines. However, the user experience, productivity and reuse may be improved with application of modern software engineering practices.

In this paper, the effect of domain level graphical modeling to reuse and productivity has been discussed with references to REFA and SMP2. It's obvious that the characteristics of simulation infrastructure is very important in the overall simulator quality, however intellectual organization of knowledge concerning the satellite systems is a more critical task and needs as much attention. Yet SMP2 alone does not cover the problems TAI faced in its own development efforts using the traditional SMI interface. While SMP2 supports compatibility and interoperation between different phases of the mission, like integrating programming languages and enabling cross-platform operations, application of REFA will mostly enhance reuse and interoperability between missions.

While it is possible to use UML notation techniques for conceptual modeling of simulated systems and generating/merging source code to acquire the final software, this process is not required to be repeated for each mission. It may be more convenient to use a graphical modeling environment to compose the simulators from software/hardware components using the guidelines presented in this work. The proposed method requires mappings between the domain level links and software infrastructure used.

As a final note, reference architecture should be supported by publicly/commercially available tools and libraries that endorse the applicability of the architecture. The architectures or standards which do not have variety of accessible implementations and technical support contain risk factors when evaluated be utilized by industries.

REFERENCES

[1] L. Argüello, J. Miró, J.J. Gujer and K. Nergaard, “SMP: A Step Towards Model Reuse in Simulation”, *ESA Bulletin 103*, 2000.

[2] H.Luban, D.Hincu, “Interdependency Between Simulation Model Development And Knowledge Management”, *Theoretical and Empirical Researches in Urban Management* , vol .4, pp. 75-85, 2009.

[3] V.Rodrigues, “On the Specification of Spacecraft Simulators using Object-Oriented Methodologies”, MS Thesis, University of Porto, 2007.

[4] Y.Lei, L.Song, W.Wang and C.Jiang, “A Metamodel-Based Representation Method For Reusable Simulatıon Model”, *Proceedings of Winter Simulation Conference*, 2007.

[5] I. Rus, M. Lindvall, “Knowledge Management in Software Engineering”, *IEEE Software*, vol. 19, pp. 26-38, 2002.

[6] A. Goldfinger et al. , “A knowledge-based approach to spacecraft distributed modeling and simulation”, *Advances in Engineering Software*, vol. 31, pp. 669-677, 2000.

[7] V. Reggestad , N. Sebastiao, and J.Pereria, “Improved concept for database integration in Operational Simulators: A key to model reuse”, *Proceedings of AIAA SpaceOps 2010 Conference*, 2010.

[8] N.Sebastiao , “A Reference Architecture for Spacecraft Simulators”, *Proceedings of AIAA SpaceOps 2008 Conference*, 2008.

[9] H.T.Pham et al. , “An Industrial Validation of SMP”, *Proceedings of 10th SESP Workshop*, 2008.

[10] ESA. “SMP 2.0 Handbook”, *Technical Report*, *EGOS-SIM-GEN-TN-0099*, 2004.

[11] Y.Lei, W.Wang, Q.Li, Y. Zhu, “A transformation model from DEVS to SMP2 based on MDA”, *Simulation Modelling Practice and Theory*, vol. 17, pp. 1690-1709, 2009.

[12] P. Cobas-Herrero, B.G. Gutiérrez, R.P. Vara, R.A. Rodríguez and C.G. de la Malla, “An ESA State-of-the-Art Simulation Tool for Space Applications”, *Proceedings of 9th SESP Workshop*, 2006.

[13] X.Cyril, “Rosesat - A Graphical Spacecraft Simulator For Rapid Prototyping”, *Proceedings of AIAA SpaceOps 1998 Conference*, 1998.

[14] SMProcess, “FxEngine Framework User Guide”, http://www.smprocess.com/Docs/FxEngine.pdf [retrieved 21 August 2010].

[15] M.Pignède, J.Morales, P.Fritzen and J.Lewis, “Swarm Constellation Simulator”, *Proceedings of AIAA SpaceOps 2010 Conference*, 2010.

[16] P.Shames, T.Yamada, “Tools for Describing the Reference Architecture for Space Data Systems”, *Proceedings of Ground System Architecture Workshop*, 2004.

[17] H.Praehofer, J.Sametinger and A.Stritzinger, “Building Reusable Simulation Components”, *Proceedings of WEBSIM 2000, Web-Based Modelling & Simulation*, 2000.

[18] S.Jeong, S.Lee, J.Bang, J.Kim and T.No, “Generalized Satellite Multi-Body Attitude Dynamics Modeling using Component-based Design Methodology”, *Proceedings of 25th AIAA International Communications Satellite Systems Conference*, 2007.

[19] Physics Abstraction Layer, http://pal.sourceforge.net, [retrieved 21 August 2010].

[20] Scythe Physics Editor, http://sourceforge.net/projects/physicseditor, [retrieved 21 August 2010].