Model Driven Development for Distributed Simulation using SysML

Michel Keuning, Arno Gerretsen
National Aerospace Laboratory NLR
Amsterdam, The Netherlands
keuning@nlr.nl, agerretsen@nlr.nl

ABSTRACT

How to build joint distributed mission simulations that are more effective with respect to the set objectives? Faced with already available simulators, optimal matching between these simulators in a distributed mission simulation is normally not possible. Often however these simulators can be connected together and configured such that they have at least basic interactions in a common environment. This however gives rise to questions on the effectiveness of such distributed simulations, which primarily depends on the objectives set out for the distributed simulation. When a simulation can be effectively used to satisfy an objective it is said to have effective realism with respect to this objective.

Interoperability standards for distributed simulation, first Distributed Interactive Simulation (DIS) later High Level Architecture (HLA), focus on information distribution. However, these interoperability standards do not specify how distributed information shall be used within the receiving simulations, neither do they specify to what degree reality must be modelled. Both well defined information usage and an adequate abstraction of reality throughout the distributed simulation are key interoperability factors to ensure effective realism.

This paper proposes a method, named Model Driven Development for Distributed Simulation (MD3S), to ensure better effective realism of distributed simulations. This method supports the Federation Development and Execution Process (FEDEP) and incorporates objectives, requirements, constraints, scenario interactions, conceptual model, design and some implementation aspects into a single unified and fully correlated development model, which also benefits verification and validation. It is based on common systems engineering practices and uses the standard Systems Modelling Language (SysML) for model expression. This paper presents the MD3S concepts by means of a case-study that has been performed.

ABOUT THE AUTHORS

Michel Keuning is a senior scientist with the NLR, working at the training, simulation & operator performance department. His career at NLR began in 1996 in the field of safety critical software for airborne systems; by the turn of the century he had moved into the field of human-in-the-loop real-time simulation with focus on distributed mission simulation. He holds an MSc in Computer Science from the University of Twente.

Arno Gerretsen is an R&D engineer with the NLR, working at the training, simulation & operator performance department since 2005. His areas of work include modelling of simulation software, computer generated forces and the visual systems of the flight simulators. He has an MSc in Aerospace Engineering from the Delft University of Technology.

Model Driven Development for Distributed Simulation using SysML

Michel Keuning, Arno Gerretsen
National Aerospace Laboratory NLR
Amsterdam, The Netherlands
keuning@nlr.nl, agerretsen@nlr.nl

INTRODUCTION

How to build joint distributed mission simulations that are more effective with respect to the set objectives? Faced with already available simulators, optimal matching between these simulators in a distributed mission simulation is normally not possible.

Often however these simulators can be connected together and configured such that they have at least basic interactions in a common environment. But that is usually a costly process, both in terms of time and money required, and it remains a question whether the resulting distributed simulation can effectively meet the objectives set out for it.

This paper proposes one means of improving development of effective distributed simulations: a unified model driven method to create systems engineering models that merges distributed simulation specific standards with standards and best practices from the domains of systems and software engineering. Systems engineering models are similar to the blueprints of a building. The reason engineering models are built is to better understand and have more control over the system under development, thus ensuring that the result is more effective to its intended use.

Currently, there is no general agreement on one method to produce engineering models for distributed simulations that covers all of the development process. Rather, the various stages of development are supported by dedicated methods and resulting engineering models. The work that is most closely related to the presented work is that on conceptual modelling, especially those that adopt formal modelling languages like the Unified Modelling Language (UML) as basis for conceptual modelling.

The method proposed in this paper, called Model Driven Development for Distributed Simulation (MD3S), is used to produce a unified engineering model. MD3S takes the use of UML for conceptual modelling a step further by combining SysML, an enhanced version of UML specific for systems engineering, with the concepts of the Model Driven Architecture (MDA) to cover all steps of the development process up to and including implementation.

The development of MD3S is part of an encompassing research that is focussed on how to achieve 'effective realism' in distributed simulations. The encompassing research is performed under authority of The Netherlands Ministry of Defence by a cooperation of the national research institutes NLR and TNO.

CONTEXT

The research programme that supports the development of MD3S deals with the question on how to build distributed simulations with effective realism. Not all simulations that are available for a mission simulation are created equal. This leads to the problem of fair play and the resulting doubt of usage validity. The degree to which simulations can be used together depends on the intents and objectives with which the distributed simulation is being created. Visual capability disparities for example may not be relevant in a beyond-visual-range engagement, but they certainly are when getting up close and personal.

The use of multiple simulations that cooperate to realise common objectives requires carefully engineered interoperation. Effective realism in this context is realised when the distributed simulation is an adequate abstraction of reality with respect to the set objectives. By definition this means that a networked simulation that is effective for one objective is not necessarily effective for another objective, even when the objectives seem similar.

MD3S is proposed as one means to enhance effectiveness of distributed simulations by means of a more formal development approach, tightly relating objectives to conceptual model, design and implementation. The intent is to ensure that the proper interactions are implemented and that those interactions are properly handled by each of the participating simulations.

One concept that is important to the thought behind MD3S is that of Levels of Conceptual Interoperability Model (LCIM) as formulated by Dr. Tolk (2003). The LCIM defines five levels of interoperability, in summary:

Level 0. System Specific Data – No interoperability.

Level 1. Documented Data – Interfaces are well defined (it is documented which language each one speaks).

Level 2. Aligned Static Data – Interfaces are aligned such that they match (we agree to use a common language).

Level 3. Aligned Dynamic Data – The use of exchanged information is well defined (when we send exchange information, it is know how we will deal with it).

Level 4. Harmonised Data – The sub-set of reality that is modelled and its impact on the distributed simulation are well defined (our capability envelope is well defined and harmonised for the application objective).

Interchange protocols get as far as level 2 and some technologies are developed to cope with a subset of level 3 issues. However, aligning dynamic data between simulations depends largely on adequate modelling of dynamic aspects. Dr. Tolk states that level 4 cannot be achieved by means of technology; the conceptual model is the only means to harmonise data to achieve level 4 interoperability. And level 4 is the level needed to make conclusive statements about effective realism of a distributed simulation.

We agree with this statement made by Dr. Tolk and thus regard the conceptual model the key part of MD3S to address the main research question of how to achieve effective realism. As such this paper will focus on the engineering process from the beginning up until the conceptual model. The latter stages of design and implementation are also regarded important parts of MD3S, but are not the key focus of the encompassing research and are elaborated at a later stage.

PROCESS, METHOD, MODEL, AND LANGUAGE

It is important to separate the notions of process, method, model, and language from each other. This section establishes a common ground of understanding for what is meant by them, how they relate and what are the commonly accepted standards in the community of distributed simulation engineering.

A process describes the steps to be taken and what to do along the way of each step to have ones developments performed in a controlled way. The processes intent is to guide development, such that product quality and productivity are optimised.

An engineering method on the other end is a recipe to support the activities defined in the development process, it is a means to capture the ‘how’. Such recipe prescribes how to create descriptions from which a real system can be created: the engineering model.

This model is a coherent description of a distributed simulation, used for engineering of such simulations and containing all levels of detail applicable to the engineering of the simulation. This means that there is used only one coherent interrelated description for the whole development cycle. The description will be continuously refined and extended during development and contains multiple levels of detail that however will be interrelated with each other. For this paper the word model will refer to an engineering model as defined above. If other kinds of model are being referred to, like e.g. simulation models, it will be explicitly state so.

The model is created in a language, the modelling language. This language can be a natural language or formal language. Whereas natural languages are easy to understand for humans, they are also inexact (often more than one interpretation is possible) and are less suitable for conveying structure. There exist formal languages for a very wide scope of different purposes. Mathematics, programming languages, mark-up languages are all examples of formal languages. Also there exist various specialised modelling languages, of which the Unified Modelling Language (UML) is probably the best known.

The current practice for distributed simulation development is that there exist standards for process, the Federation Development and Execution Process (FEDEP), and interaction, Distributed Interactive Simulation (DIS) and High Level Architecture (HLA).

There is currently no such agreement is on method and engineering language. There are ongoing efforts to define method or language for parts of the process, for example the Simulation Conceptual Model (SCM) and for scenario definition, the Military Scenario Definition Language (MSDL). Some, like the SCM, are still in an

investigation stage. MSDL on the other hand is being developed.

Besides the FEDEP as the development process of choice, there are two other key enabling technologies behind MD3S: the SysML modelling language and the Model Driven Architecture (MDA).

THE SYSML MODELLING LANGUAGE

This paper proposes to use one unified model across development, right up until the implementation. To facilitate creating such model a language is needed that provides all means necessary to create the unified model.

In the software engineering world unification of engineering model and language are being enabled by the Unified Modelling Language (UML). UML can be considered a mature, but continuously improving, modelling language.

Constructing distributed simulations is more than software engineering, it is very much like systems engineering. Software engineering concepts can be used as a basis, but have to be extended to cover the specific needs of systems engineering. 2006 Brought just that, the Systems Modelling Language (SysML), a tuned and extended UML specifically designed to cover systems engineering needs. Like UML, it is a language and no method or process is provided.

Of course there are more systems engineering languages around. However since SysML is fully supported by the International Council On Systems Engineering (INCOSE) and tool support is already readily available from multiple vendors, it is expected that SysML will quickly become the industry standard modelling language for systems engineering. In the past we have seen the same happening with UML for software engineering. Another direct advantage is that, since distributed simulations are software heavy, the systems engineering process can seamlessly connect to software engineering for certain parts of the simulation.

The model, described with SysML, is the blueprint of the distributed simulation, but who can understand the model? Like with any language, one has to learn the language before being able to understand it. Introducing a new language is for this reason something that should be done with great caution. Who in the community of distributed simulation 'speaks' the language? What about those that do not? Do we need to teach and preach the language to every single person involved?

For practical reasons it is not feasible and desirable to teach each and every involved person SysML. Not everyone has the proper background and one cannot expect such effort from a customer. But neither is it needed to teach every involved person whole of SysML. For most people it suffices to understand a small portion of SysML. For example requirements diagrams need to be understood by end-users, but luckily those are easy to understand.

For those parts of the model where SysML does not provide an adequate representation that fits with the background of the people that need to understand those models, UML and SysML offer the flexibility to alter notation to better fit with the application domain.

MODEL DRIVEN ARCHITECTURE

Another important background for the development of MD3S is the Model-Driven Architecture (MDA), which is an architecture developed by the Object Management Group (OMG) to support software development. A quote from the *MDA Guide (2003)*:

> *The Model-Driven Architecture starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform.*
>
> *MDA provides an approach for, and enables tools to be provided for:*
>
> - *specifying a system independently of the platform that supports it,*
> - *specifying platforms,*
> - *choosing a particular platform for the system, and*
> - *transforming the system specification into one for a particular platform.*
>
> *The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns.*

The basic concept behind the MDA defines three distinct viewpoints:

1. Computation independent viewpoint; represented by a Computation Independent Model (CIM). The CIM is a domain model

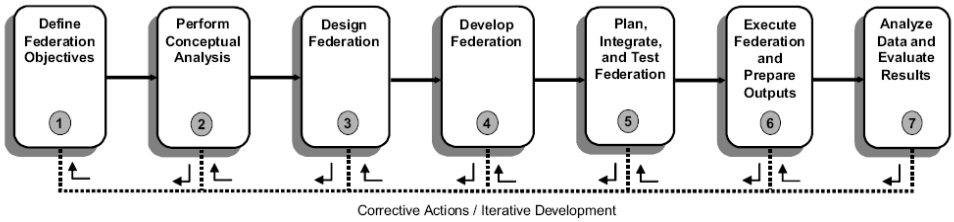

Figure 1: Steps of the FEDEP

with the purpose of bridging the gap between domain experts and solution experts.

2. Platform independent viewpoint; represented by a Platform Independent Model (PIM). The PIM is a model that describes the operation of a system without the inclusion of details that are specific to a particular solution.

3. Platform specific viewpoint; represented by a Platform Specific Model (PSM). The PSM is a translation of the PIM to a specific platform; it is a direct representation of an implementation.

UNIFICATION

The FEDEP is taken as the process basis for MD3S. It is the industry accepted standardised process to develop distributed simulations and it is periodically updated to reflect the latest insights. A model driven engineering approach does not need a different process; it is a different way of implementing the process. The FEDEP combined with MDA fundamentals and the SysML language form the basis of MD3S.

SysML is used to express all of the MD3S models, meaning that there is no mismatch in expression language between the stages of development. The matching of the MDA architecture fundamentals to distributed simulation development and the FEDEP is as follows (see Figure 1 for the steps of the FEDEP):

- The MDA Computation Independent Model (CIM) is matched to FEDEP step 1, the definition of federation objectives. The result is an objectives model that represents the viewpoint of the application domain expert.

- The MDA Platform Independent Model (PIM) is matched to FEDEP step 2, the conceptual analysis. The result is a set of usage scenarios and a conceptual model annotated with additional federation requirements.

- The MDA Platform Specific Model (PSM) is matched to FEDEP steps 3, the design. The result is a design that is specific to the selected platforms and other resources.

Referring back to the levels of interoperability, it is clear that the specification of the PIM including full traceability back to the CIM is the crucial step of engineering a distributed simulation with respect to insurance of adequate effective realism. The PSM is a linear transformation of the PIM to a specific set of target platforms.

In practise it often happens that the specifications in the PIM cannot be satisfied to 100% by the selected resources. In this case round-trip iterative engineering should be applied. The available full traceability provides a powerful means to perform reverse analysis of the impact on the objectives of the resource constraints. Based on this analysis it is possible to make well founded decisions to change objectives. These changed objectives result in changes to the PIM through normal process flow.

MD3S: ILLUSTRATED BY CASE STUDY

A case study is being elaborated in support of the ongoing research into effective realism of distributed mission simulations. MD3S is applied to this case study, which provides a feedback on the usability of the method and initiate improvements. The setting for this case has been chosen to be a Close Air Support (CAS) scenario in which a forward air controller (FAC) team on the ground and a flight of two F-16 fighters together perform a training task in a virtual mission environment. This section describes the MD3S method illustrated with examples from this case study.

Case description

The main objective of the CAS training case is that both the fighter pilots and the FAC-team get training in following the correct procedures while delivering a laser guided weapon on a ground target. An additional objective is that the target will be laser designated by

the FAC-team from the ground, instead of by the fighters from the air. Also no advanced technologies, like an automatic target handover system (ATHS) or a targeting pod video-link shall be used during this training. So it can be said that this is a training of the basic procedures using voice communication interactions between the ground and the air only.

Objectives analysis

Following the prime objective of the case, as specified above, a further objectives analysis is required to be able to derive the requirements for all the components that form the distributed mission simulation.

Requirements engineering as such is not new and therefore the industry already has its common practices, standards and tools to support this part of the development process. A tool such as DOORS has become the de facto standard for example. This gives rise to the question why to do the requirements engineering in a different way using SysML?

The rationale for integrating requirements engineering in the MD3S engineering model is to improve correlation of requirements with their derived model elements. The traceability of the requirements during the development process is also improved. Tool support exists that allows connecting SysML and current industry standards, thereby combining the benefits of both worlds.

This part of MD3S maps onto the 'Define Federation Objectives' phase of the FEDEP. But whereas the FEDEP has identified two separate steps, 'Identify User/Sponsor Needs' and 'Develop Objectives', for MD3S it has been decided to merge the result of those steps into one single objectives model. The development of objectives is regarded as an elaboration of the user needs. The use of a single objectives model improves consistency and prevents duplication of requirement formulations.

Based on the activities as identified in the FEDEP certain categories of objectives have been identified for MD3S. Examples of these are critical system, fidelity, security or scenario objectives. But also resource availability and evaluation criteria can be addressed in this phase. Which kind of objectives need to be derived exactly also depends strongly on the aim of the distributed simulation being constructed.

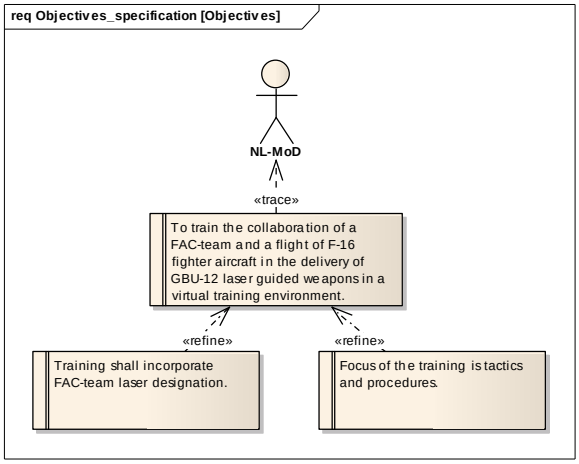

Figure 2: Example user needs specification

Looking at the CAS training case this means that the main objective as outlined before can be expressed in a SysML requirements diagram. Actors are used to clearly track the origin of an objective. The requirements that are related to an actor and their refinements are the original user needs as defined in the first sub-step of the FEDEP. The requirements that are identified as derived are the objectives from the second sub-step of the FEDEP. In that way the difference between those types of requirements still remains identifiable, although all requirements are combined into one model. Specification of user needs is illustrated in Figure 2, while Figure 3 illustrates the derivation of objectives.

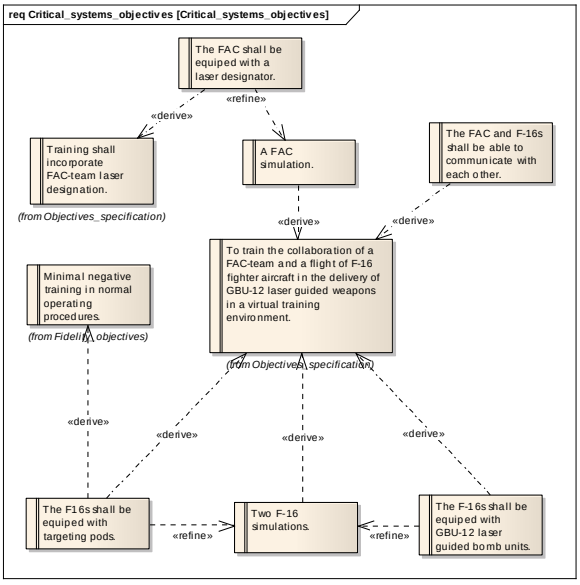

Figure 3: Example objectives specification

Conceptual analysis

The FEDEP defines three sub-steps for its second step, the conceptual analysis. These sub-steps are:

1. Develop scenario
2. Develop federation conceptual model
3. Develop federation requirements

MD3S adopts two perspectives in its PIM to cover the products of all three FEDEP sub-steps:

1. Scenarios
2. Conceptual model

The target of the MD3S scenarios is to specify the following three aspects for a distributed simulation:

- Specification of the context of the simulation by means of identification of the actors that interact with the simulation.
- Specification of the use cases of the simulation.
- Specification of the scenario flows for each of the use cases.

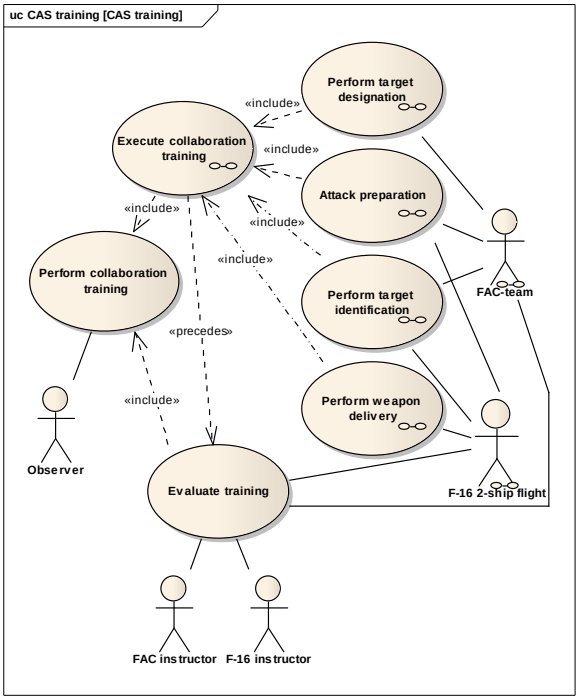

Figure 4: Example use case specification

Scenario specifications are modelled by means of use case diagrams (see Figure 4). For the detailed scenario specification either activity diagrams or sequence diagrams are used (see Figure 5). In those the different activities can be identified and assigned to the relevant player roles.

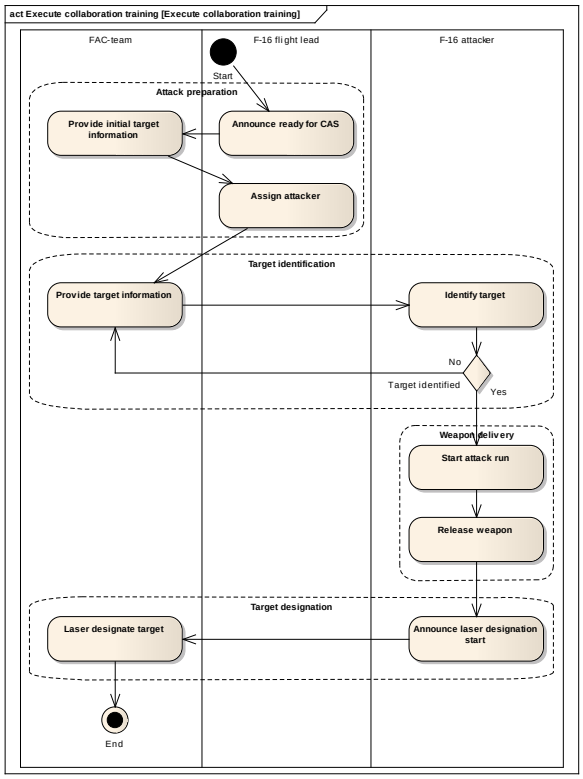

Figure 5: Example scenario specification

The sub-steps for federation conceptual model and federation requirements are combined into one, whereby the federation requirements become an integral part of the conceptual model and are directly related to the conceptual model elements.

The target of the MD3S conceptual model is to specify the following four aspects for a distributed simulation:

- Specification of significant distributed simulation elements and their associations.
- Specification of the dynamic interactions between associated simulation elements.
- Specification of the required behaviour of simulation elements as a result of interactions.
- Specification of the relation between real-world elements and their simulation counterparts.

The latter bullet is specific to the domain of modelling and simulation in that the intent of a simulation is by definition to represent real-world elements in a simplified form. The degree to which this simplification is done is a crucial aspect in realising adequate effective realism. This is also the most difficult part to specify in the MD3S model.

MD3S takes the approach to supply a component library that contains SysML descriptions that represent real-world elements, including the relations between and the compositions of these real-world elements if applicable. The conceptual model identifies simulation elements that realise these real-world representations with the possibility to specify appropriate constraints on this realisation.

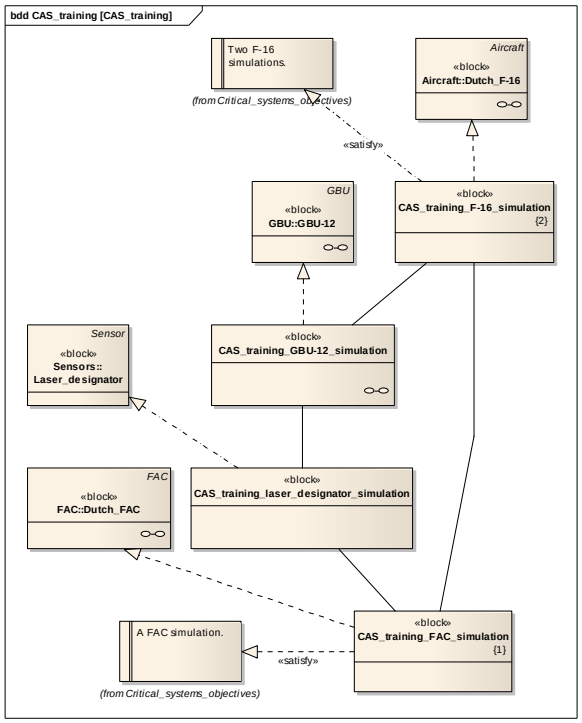

Figure 6: Example specification of distributed simulation elements, their associations and which real-world elements they implement

Specification of significant distributed simulation elements and their associations is done by means of SysML Block Definition Diagrams (BDD). In SysML the term block is an abstraction for a system or system component; in MD3S a block is an abstraction of any element that needs identification in a distributed simulation. These can be whole simulations, but also models that make up a simulation or supporting systems like data loggers. Associations between blocks indicate that there are interactions possible between the blocks. Special associations are available to specify block composition and block similarity (inheritance).

In the same block diagrams the relations to the real-world domain are specified by means of so-called realisation associations between simulation elements and elements from the component library represent the real-world elements. For further details of these real-world representatives, reference can be made to their specification in the component library. An example of such specification is presented in Figure 6.

Specification of the dynamic interactions and the resulting required behaviour of the simulation elements is done by means of SysML activity diagrams (ACT), sequence diagrams (SD), the definition of event triggers on blocks (in BDD), state machine diagrams (STM), and parametric diagrams (PAR). It is regarded as engineering freedom for the system architect to decide which combination of diagrams best suits the behaviour of the system being modelled. The important message here is that SysML provides substantial means to formally specify interaction and behaviour.

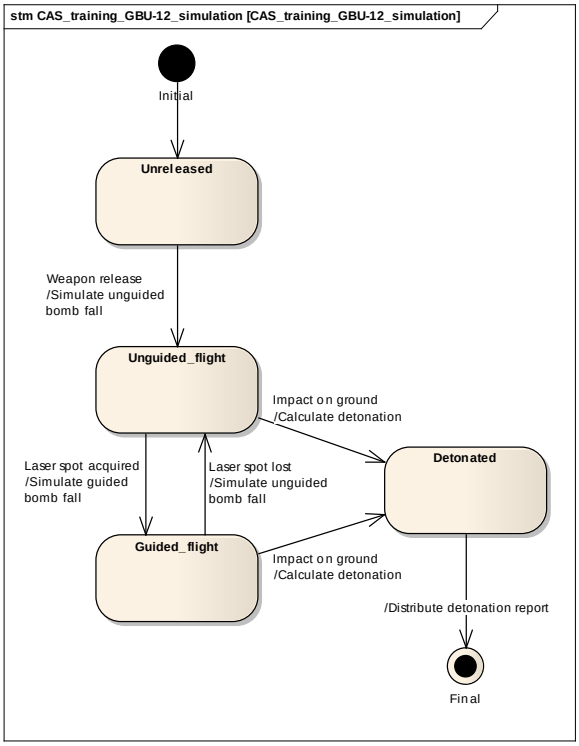

Figure 7: Example state machine diagram for a laser guided bomb simulation

Figure 7 shows an example of using a state machine diagram to specify how input events trigger action and state changes, which in turn results in output events.

The activity diagram shown in Figure 8 complements the state machine diagram of Figure 7 by specifying the activities related to the identified states of the laser guided bomb simulation.

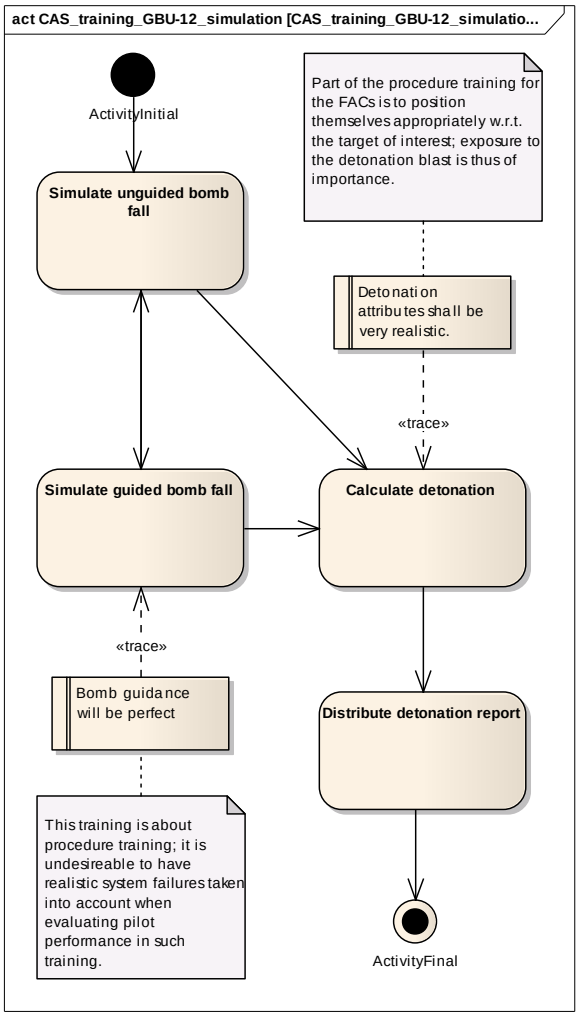

Figure 8: Example activity diagram for a laser guided bomb simulation

Where applicable it is possible to include system requirements in any of the SysML diagrams and associate those requirements to those engineering model elements where they apply to. System requirements are thus fully embedded in the conceptual model with full traceability and a coherent view for the conceptual model architects. Notes can be used to document rationale for decisions. Figure 8 includes examples of additional system requirements and notes about their rationale.

Lessons learned

While working on this case study it became very apparent that tightly integrating requirements definitions with the conceptual model greatly benefits a clear understanding on how objectives are translated into functional blocks that interact. Working on the problem with an expanding model, whereby a model on a higher level of abstraction is translated into a model on a lower level of abstraction by adding to the overall model with explicit traceability, adds to this positive effect as the steps of analysis can be clearly followed through the available traceability. SysML is a very promising formal language for such modelling approach and provides good means to model static relations, interactions, and dynamic behaviour as well as traceability and annotation.

Further maturation of MD3S is therefore regarded as desired by the authors. Within the current research programme continued improvement of MD3S is pursued and additional studies will be performed to analyse how well the approach is suited to cope with adaptations. For this, cases will be elaborated that are variations of each other so that it becomes clear how a new case can be developed from an existing case. Behaving well to adaptations is regarded as a key element to a successful method, because in the field of distributed mission simulation it is very common to have variations and extensions to earlier developed simulation exercises. With current insights it is not expected that the MD3S approach will falter on this point; quite the opposite, MD3S is expected to be a key technology to better cope with change.

CONCLUSIONS AND FUTURE DEVELOPMENTS

This paper proposes a unification of the FEDEP, the MDA modelling architecture, and the SysML modelling language into a method to engineer distributed simulations. This Model Driven Development for Distributed Simulations provides a number of benefits to the development of distributed simulations, amongst others:

- One formal engineering language baseline, i.e. SysML, which is an industry standard.
- Complete traceability integrated into the resulting engineering model.
- Requirements are not only visually related to each other, but also to model constructs in other stages of specification and development.

- More formal specification is less susceptible to misinterpretation.

- All aspects for full interoperability are taken into account, including behaviour specification and relation to the real-world elements that are modelled and simulated.

Current focus is on the initial steps of the FEDEP, which are the most important steps for the encompassing research project that is concerned with effective realism of distributed mission simulations. The concept behind MD3S is however to cover all of the simulation development cycle and the basic building blocks for that concept are already in place:

- The MDA concepts of transformation from platform independent model to platform specific model maps to the process of going from FEDEP step 2 (conceptual analysis) to FEDEP step 3 (design federation).

- SysML and UML provide the power to iterate into design level detail. Because SysML is a customisation of UML, necessary software developments become an integral part of the whole MD3S model.

- UML also allows for data modelling, which provides excellent means for integrating development of High Level Architecture (HLA) object models.

Other authors have proposed the use of UML/SysML and MDA for use in distributed simulation development for more specific application. However, the full power of these can only be achieved by full complementary integration of UML/SysML, MDA, and the FEDEP; from the first step to the last.

A case study has been performed as an initial evaluation of the MD3S concepts. This case study illustrated the benefits mentioned above and provides a first validation that MD3S is a good line of though. Although further research is needed to fine tune the method and provide practical insight in to its robustness against change.

Therefore related cases will be elaborated next. In these variation on the scenario of the first case will be made, focussing on CAS for mission rehearsal instead of training. This second case study will provide insight into change engineering aided by MD3S and the method will be enhanced accordingly.

REFERENCES

Booch, G. & Rumbaugh, J. & Jacobson I. (1999) *The Unified Modeling Language User Guide.* Reading, Massachusetts, USA: Addison Wesley Longman Inc.

Tolk, Dr. A. (2002) *Avoiding another Green Elephant – A proposal for the Next Generation HLA based on the Model Driven Architecture.* Fall Simulation Interoperability Workshop, paper 02F-SIW-004.

Lutz, L. & et. al. (2003) *1516.3 IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP).* New York, NY, USA: The Institute of Electrical and Electronics Engineers, Inc.

Keuning, M. & Lemmers, A. (2003) *RTP 11.13 gets to grips with SE specifications.* European Simulation Interoperability Workshop, paper 03E-SIW-097.

Tolk, Dr. A. (2003) *The Levels of Conceptual Interoperability Model.* Fall Simulation Interoperability Workshop, paper 03F-SIW-007.

Parr, S. & Keith-Magee (2003), Dr. R. *Making a Case for MDA.* Fall Simulation Interoperability Workshop, paper 03F-SIW-026.

Object Management Group (2003) *MDA Guide.* http://www.omg.org/docs/omg/03-06-01.pdf

Tolk, Dr. A. (2004) *Metamodels and Mappings – Ending the Interoperability War.* Fall Simulation Interoperability Workshop, paper 04F-SIW-105.

Object Management Group (2006) *OMG SysML Specification.* http://omgsysml.org/.

Hause, M. & Moore, A. (2006) *The Systems Modeling Language.* Beaverton, Oregon, USA: ARTiSAN Software Tools.

Balmelli, Dr. L. (2006) *An Overview of the Systems Modeling Language for Products and Systems Development.* IBM Research Division.

Weilkiens, T. (2006) *Systems Engineering mit SysML/UML.* dpunkt.verlag GmbH.