Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

REQUIREMENTS FOR DOMAIN SPECIFIC DISCRETE EVENT SIMULATION ENVIRONMENTS

Edwin C. Valentin

Systems Navigator
Delft, THE NETHERLANDS

Alexander Verbraeck

Faculty Technology, Policy and Management
Delft University of Technology
Delft, THE NETHERLANDS

ABSTRACT

Domain specific discrete event simulation environments are supposed to enable faster and easier model development and experimentation. Unfortunately, perceived disadvantages from simulation experts hinder the wide application of this technology. We have performed laboratory experiments and simulation studies in two different domains to learn what the difficulty of domain specific simulation environments is. The lessons that we learned from these experiments and simulation studies enabled us to formulate requirements for domain specific simulation environments, for the model constructs in these environments, for the design of these environments and for guidelines for the use of these environments in simulation studies.

1 INTRODUCTION

It is hard to perform a successful simulation study. According to Sadowski and Grabau (2000) it is time consuming and requires a lot of skills to enable to perform a good simulation study. Generic discrete event simulation environments such as Arena, Promodel and Automod require model developers to make a translation of their system into concepts such as queues and resources. This step is difficult for a lot of model developers. Most vendors of commercial generic simulation environments have observed this difficulty and provide domain specific extensions to their simulation environment to ease the development in a certain domain. For example, Arena provides templates for packaging lines and contact centers (Bapat and Sturrock, 2003 and Promodel developed a version dedicated to hospitals (Harrell and Price, 2002).

The domain specific extensions to generic simulation environments, also referred to as domain specific simulation environments, have several advantages:

- better understanding of the simulation model by problem owners, because the concepts of the conceptual model can be recognized in the simulation model (Pater and Teunisse, 1997; Kasputis and Ng, 2000);
- easy to generate new simulation experiments (Pater and Teunisse, 1997; Altiok et al, 2001);
- easier validation of the simulation model, because only the applicability of the model constructs needs to be checked and not the inner-working;
- less instances of model constructs in the simulation model, with improved overview and model management (Kasputis and Ng, 2000; Altiok, 2001).

These advantages are supported by simulation experts in several panel discussions with the theme "Future in Simulation" in the past years at the Winter Simulation Conferences (Banks et al, 2001; Diamond et al, 2002; Barton et al, 2003). In each of these panel discussions the use of domain specific simulation environments is mentioned as the next step for discrete event simulation research, but the panelists also conclude that this next step is still not taken.

During the past years, we have gained experience in developing simulation models using domain specific simulation environments (section 3) and based on these experiences we learned that domain specific simulation environments are often incomplete, hard to maintain, and model developers need to overcome initial low trust for these environments (section 4). We concluded that these issues can only be improved if domain specific simulation environments match a set of requirements (section 5). Further research will be defining a reference architecture for domain specific simulation environments that match these requirements (section 6).

2 BACKGROUND OF DOMAIN SPECIFIC SIMULATION ENVIRONMENTS

Generic simulation environments enable model developers to compose simulation models using model constructs. Model constructs are things the generic simulation environment provides to compose a simulation model, ranging from complex resources that are dragged into a simulation model to objects, functions, or procedures in coded logic. Model developers who have experience with a simulation environment know after some time how to work with the available model constructs and how to represent a system using these model constructs. However, in general the model constructs are abstract and can be used for the representation of any system. This means that many model constructs are needed to represent a specific system. A simulation model that consists of a lot of generic model constructs is therefore complex and development takes quite some time.

The difficulties of generic simulation environments are even more clear once a model developer needs to make model adjustments. System changes that seem reasonably simple to a problem owner can cause quite some difficulties to be implemented into a simulation model, because often changes need to be made at many different places in the simulation model and sometimes the structure of a simulation model needs to be severely changed in order to enable the needed simulation experiments.

Vendors of generic simulation environments observed these difficulties and introduced the ability for model developers to compose their own domain specific model constructs, or even customize the generic environment into domain specific simulation environments.

Domain specific simulation environments allow model developers to develop simulation models using model constructs that represent domain specific system elements. For example, in container terminals a simulation model can now be composed out of quays, storages and vehicles (Saanen, 2004) instead of model composition using resources and queues. The simulation model development consists now of instantiating and parameterizing particular types of container cranes into the simulation model instead of developing the detailed behavior of each crane. Clearly this is less time consuming and easier, i.e. it can be performed by domain experts that are simulation novices, rather than black-belt simulation experts.

The advantages of use of domain specific simulation environments are widely accepted, but also several disadvantages can be identified from literature. For example:

- lack of trust of model developers in model constructs of domain specific simulation environments and thus no trust in the simulation models developed using the model constructs (Balci, 1997);
- lack of understandability by model developer regarding model constructs of domain specific simulation environments, due to the need for increased functionality and system behavior (Barton et al, 2003);
- unable to develop a domain specific simulation environment that provides model constructs that represent all system specific situations within a domain (Kasputis and Ng, 2000; Diamond et al, 2002; Barton et al, 2003; Petty et al, 2003);
- difficulty to prepare a domain specific simulation environment for future adjustments, because "the desire for new system capabilities is never ending" (Davis et al, 2000; p.1588);
- lack of insight by model developers as to whether a domain specific simulation environment is suitable for representing a domain (Kasputis and Ng, 2000);
- development of invalid models by model developers, due to the use of a model construct of a domain specific simulation environment for unintended system representations, either because the domain specific simulation environment could not be used or because an incorrect model construct was selected from the environment (Page and Opper, 1999; Kasputis and Ng, 2000; Diamond et al, 2003);
- overestimation by model developers of the functionalities provided by a model construct of a domain specific simulation environment (Pater and Teunisse, 1997);
- incorrect extension of the domain specific simulation environment with new model constructs by developers of domain specific simulation environments (Barton et al, 2003);
- lack of freedom for model developers in developing experiments with simulation models constructed in domain specific simulation environment (Sol, 1982);
- development by incapable simulation experts that are not sufficiently aware of the capabilities or concepts of a simulation environment.

These disadvantages are observed by many simulation experts, and are a reason to keep on performing simulation model development using generic simulation environments, to enable them to keep control over their own models. Unless these disadvantages are covered by domain specific simulation environments, the majority of the simulation models will be developed in generic simulation environments and modelers will continue to use low level model constructs.

3 OUR EXPERIENCES WITH DOMAIN SPECIFIC SIMULATION ENVIRONMENTS

The extensive experiences that we have with domain specific simulation environments have been gathered by performing laboratory experiments with simulation novices and simulation experts (Valentin et al, 2003a & 2003b), and by carrying out simulation studies in two different domains, Automatic Guided Vehicles (Heijden et al, 2002) and passenger logistics at airports (Verbraeck and Valentin, 2002).

The laboratory experiments have been performed to compare the use of a domain specific simulation environment with the use of a generic simulation environment in a simulation study for public transport. This laboratory experiment has been performed twice, once with novices in simulation and once with simulation experts.

The simulation studies in the two different domains concerned the complete cycle from development of a domain specific simulation environment to the execution of simulation experiments with simulation models that had been composed using domain specific model constructs. Within the domain of automatic guided vehicles one extensive, multi-year simulation study has been performed. In the domain of passengers at airports three simulation studies have been performed at two different airports.

The domain specific simulation environments for the laboratory experiments and the simulation studies have been developed by a team of simulation experts that were sufficiently aware of the concepts of the simulation environment and had several years of experience in software development and simulation model development. Therefore, the problems of section 2 related to inexperienced model developers have been avoided.

3.1 Laboratory Experiments

Both laboratory experiments (Valentin et al, 2003a & 2003b) deal with the development of a simulation model for the introduction of a new transportation line in a public transport system for the city of The Hague in The Netherlands. The new transportation system should connect a new urban area with the center of the city. The participants in the laboratory experiments were requested to provide insight to the problem owners in different layout and configuration possibilities. They should develop a simulation model and perform several simulation experiments with alternative layouts and different types of vehicles. Figure 1 shows the route of the new transportation system.

The participants of this laboratory experiments were divided into two groups of equal size. Half of the individuals used the generic simulation environment Arena and the other half used a domain specific simulation environment, specifically developed for modeling public transportation systems. The participants in the laboratory experiments were simulation novices, i.e. students of Delft University of Technology, and simulation experts, i.e consultants and software developers of the Arena team of Rockwell Software. Both the simulation novices and the simulation experts worked were divided over the groups that had to use the generic and domain specific simulation environment. Each participant had to develop the model individually within a limited time frame.

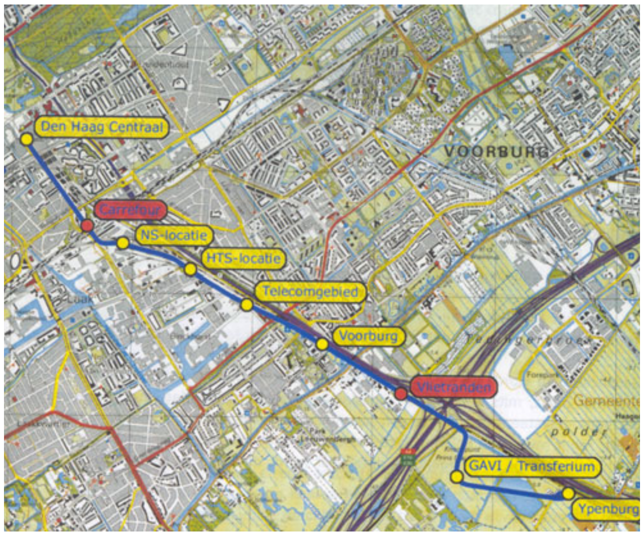

Figure 1: Layout Transportation System in The Hague

The expectations were that the participants working with domain specific simulation environments could easily compose a simulation model and provide the problem owners with a lot of insight into the system. We expected that the participants working with the generic simulation environment would develop a very abstract simulation model, only with the key processes and would provide insight only in the key performance indicators. This model would take a little bit longer to develop, but thanks to the more abstract level it was expected that simulation experiments could be performed as well.

Table 1 below shows how far the simulation experts and novices succeeded in carrying out the different steps of a simulation study. After 8 hours of working, the maximum time available for the participants in the laboratory experiments, the simulation models were analysed by observers and compared with activity logging provided by participants. It turned out that the participants with the generic simulation environment did not make any effort to reduce the level of detail of their simulation models. Therefore, these participants were far from a complete simulation model. Analysis of the simulation model showed that the novices were farther from completion than the simulation experts.

Even more surprising was that only one of the 24 participants using model constructs of the domain specific simulation environment was finished with running experiments. It appeared that the participants using the domain specific simulation environment had difficulties to make a valid simulation run. They had difficulties to get the model running and once it was running their simulation model represented invalid system behavior. The participants that were novices in simulation thought that they made a mistake and double checked all the input parameters of the instantiated model constructs. The simulation experts expected that something was wrong with the model constructs and explored the details of the model constructs and their source code to find out what they could do to get the simulation model working.

Table 1: Project Phase of Participants at End of Laboratory Experiment

Activity	**Novices**		**Experts**	
	Generic	Specific	Generic	Specific
Conceptualization				
Specification	7	1	4	
Validation		5		3
Experimentation		2		1
Finished		1		
Total:	**7**	**9**	**4**	**4**

3.2 Domain Specific Simulation Environment for the Automatic Guided Vehicles project

In 1997, the Dutch government triggered a large research effort for an advanced and highly automised cargo transportation system between the airport Schiphol and the flower auction Aalsmeer in the Netherlands. This transportation system should consist of almost 400 Automatic Guided Vehicles (AGVs) that can drive around in a dedicated infrastructure. An important factor in this study was the use of an advanced control system called TRACES (Evers and Koppers, 1996) that provided an efficient and flexible safety mechanism for the AGVs.

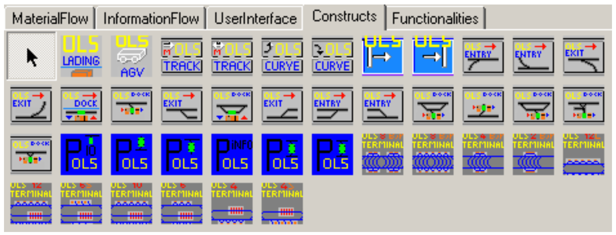

Figure 2: Model Constructs of Domain Specific Simulation Environment for AGVs

Figure 2 provides an overview of model constructs that have been developed as part of the domain specific simulation environment for AGVs. The domain specific simulation environment that has been developed for this simulation study included the main functionalities of the TRACES concept, advanced model constructs for driving behavior of AGVs and a wide range of infrastructure elements to enable configurability of terminals for loading and unloading of AGVs.

Within the simulation study over 100 experiments have been performed and compared. These experiments concerned different layouts of a terminal for loading and unloading AGVs, different mechanisms for parking or physically unloading the AGVs and control procedures for allocating new orders to AGVs or a docking position to AGVs. The result of these simulation experiments were 30 guidelines that other participants in the project could use for the design of the terminals, the control mechanisms and the AGVs (Verbraeck et al, 1998; Heijden et al., 2002).

Although the developed domain specific simulation environment was a big help in the rapid development of models, it also limited us in the performance of simulation experiments. In some cases errors occurred during the simulation run when we had replaced one or more model constructs. In other cases we were not able to provide the correct value for performance indicators due to the way we calculated values. Finally, we tried to train two other simulation experts to use the domain specific simulation environment to enable performing more and faster simulation experiments, but it took almost a month of full time hands-on training before the simulation experts were able to offer their support to the project.

3.3 Domain Specific Simulation Environment for the Passengers at Airports Project

The management of airports all over the world deals with issues like increasing the number of airplane visits, more passengers and changing procedures for passengers within the terminal. In addition the security levels have been raised enormously and therefore the flexibility for passengers that are late for a flight has decreased. Nevertheless, airport management wants to provide a maximum level of service to passengers and enable passengers to spend as much time as possible to shop or entertain themselves during their airport stay.

Amsterdam Airport Schiphol expected a growth in the coming 10 years from 40 million passengers to 60 million passengers a year. They wanted to know what they needed to change their infrastructure to accommodate this increasing number of passengers at their airport. In the past, several simulation studies had been performed at Schiphol airport, all of them provided valuable feedback, but also none of these simulation models could be adjusted well enough to the new questions, simply because this would be too complex and time-consuming.

This observation by Schiphol was a trigger to develop a domain specific simulation environment for passengers at airports (Verbraeck and Valentin, 2002). This simulation environment contains check-in counters, security checks, passport checks, areas such as walkways, lounges, sitting areas, and gates to represent the infrastructure, and passengers that claim and use this infrastructure. These different types of infrastructure were all represented by a specific model constructs. Figure 3 shows a part of the hierarchy of the model constructs for hierarchy as they were available in the domain specific simulation environment.

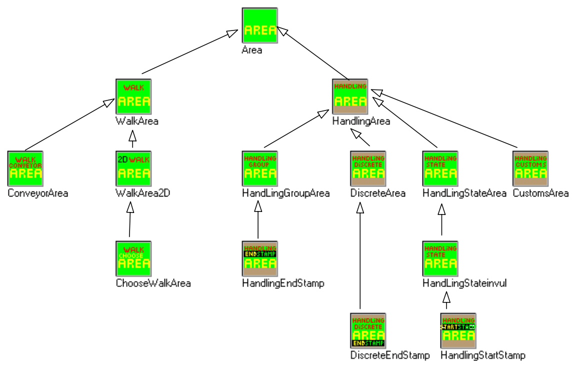

Figure 3: Infrastructure Model Constructs in Domain Specific Simulation Environment for Airports

The model constructs of the domain specific simulation environment have been used several times for different simulation studies. The modeling of the airport Schiphol was the first project where the domain specific simulation environment has been applied. In this simulation study the complete airport has been modeled including all paths passengers can take to reach their destination. The simulation model has been used to experiment with different extensions in the layout of the airport and different ways of allocating airplanes to gates (Arends, 1999). Two other simulation studies focused only on the processes of check-in. We have modeled the departure hall of the KLM at Schiphol airport including the growing use of e-tickets and internet bookings of the KLM and the optimum way of allocating airlines over the limited available check-in desks in the new international terminal of JFK (Valentin, 2003).

All three simulation studies provided insight to the problem owner, but the projects at Schiphol and JFK required a long lead-time before the results could be presented to the problem owners. The main cause of the delay was the development of specific model constructs. New model constructs had to be developed during these simulation studies, including debugging, which took quite some time.

4 LESSONS LEARNED

4.1 Lesson Learned from Laboratory Experiments

The laboratory experiments showed that model developers using a domain specific simulation environment achieve more results within 8 hours than model developers who start with a generic simulation environment with the same time limit (see Table 1). ‘More results’ refers to simulation models that are closer to be ready for experimentation. The model developers using model constructs from a domain specific simulation environment were able to carry out more simulation experiments, delivered simulation models that were better understandable, easier to extend and were more satisfied about the usability and quality of their work. Furthermore, the model developers expect that they would need less time to perform a subsequent simulation study now that they had a good working knowledge of the domain specific simulation environment.

The laboratory experiments also taught us that the importance of model developers’ understanding of model constructs had been underestimated. The different participants in the laboratory experiments, i.e. novices and simulation experts, indicated that the type of documentation and training given before starting to use a domain specific simulation environment should be tailored to fit the type of simulation model developer. Experts need to be convinced that the model constructs of the domain specific simulation environment are valid and that they still allow for freedom and choice, before allowing them to work with the domain specific simulation environment. The main thing that these technical experts wanted, was deeper insight into the model constructs of the domain specific simulation environment. They saw the domain specific simulation model constructs as black boxes and they did not fully accept what was going on. The novices, who had few prior prejudices to overcome, worked directly with the model constructs with only a few problems regarding the use of model constructs, but they needed to be coached within the modelling process itself. A structured process for using a domain specific simulation environment for developing a simulation model might be very helpful to overcome their problems. Also clear support and examples in case of errors like a frequently asked question list seems to be needed.

4.2 Lessons Learned from Case Studies

The four simulation studies that have been performed in the two domains provided us with insights in the use of domain specific simulation environment to compose simulation models. We divide the lessons learned during the simulation studies in the two domains with relation to the use of model constructs and the development of model constructs. In the next section we will use these lessons as a starting point for the requirements for domain specific simulation environments, for design approaches for model constructs of domain specific simulation environments and for guidelines for use of domain specific simulation environments in simulation studies.

4.2.1 Lessons Learned Regarding the Instantiation of Simulation Models

Model developers can easily understand model constructs that represent simple physical system elements. This understanding is reduced when the complexity of the model construct increases. Examples of simple physical system elements are the track (AGV) or area (Airports). An example of a physical system element with a larger number of processes is the check-in counter model construct with all its variants (Airport).

Model composition is easier when separate model constructs are available to represent infrastructure system elements and process descriptions. E.g. tracks and scripts for AGVs to use these tracks, or areas and scripts for passengers to describe what to do at airports.

Insufficient testing of model constructs by model developers reduces the confidence that model developers have in the domain specific simulation environment and enlarges the project lead time, because model developers can not solve issues that appear during testing. E.g. development of new model constructs for airport that are directly applied in the Schiphol project (Airport).

A large set of model constructs that all represent the same system element at different levels of reduction is hard to use, because model developers have difficulty in selecting the most suitable model construct. E.g. selecting of appropriate check-in areas in Schiphol and JFK project (Airport).

Output reports of domain specific simulation environment that creates an information overload of a dump of performance indicators that are gathered by all instances of model constructs in the simulation model are not usable for a model developer. E.g. our simulation model for the Schiphol project in which each model construct of an area provided detailed performance indicators. A suitable alternative was developed in the AGV project where a selected set of statistics was represented with advanced graphical outputs.

Model developer tend to use only the animation of the model constructs in the simulation model. Therefore the animation of these simulation models is not any better than the animation of the model constructs. For example, The AGVs in the AGV project are represented in detail, resulting in understandable animation of the simulation model. In the Schiphol-project the animation of areas was nothing more than coloured squares, because the model constructs of an area only provided this animation.

Replacing model constructs to adjust a simulation model between two experiments often causes technical problems due to connections between instances of model constructs in the original simulation model. It turned out that often model constructs had direct pointers and links to the replaced model constructs. This leads to errors and problems during the experimentation when alternative constructs are inserted in the model.

The learning curve that experts in a domain have to follow before they can use or develop a simulation model is high. This refers to the long training period necessary for the simulation experts to be available within the AGV project.

Model developers often make mistakes in user input. The simulation models that the model developers developed often did not work exactly as they expected. Often this was caused by mistakes in parameter settings in one or more model constructs of the simulation model.

4.2.2 Lessons Learned Regarding the Development of Domain Specific Simulation Environment

Control mechanisms are hard to define in model constructs. E.g. a virtual AGV manager that allocated the AGVs to locations where the unloading or loading of the AGVs could be performed was a necessary construct, but how this model construct should perform the tasks was difficult to model in a flexible and adjustable way.

Tendency to extend features and performance indicators available in model constructs. The initial model constructs will be adjusted to include new functionalities. For example, in the project of the automatic guided vehicles the number of performance indicators that gathered data during a simulation experiment doubled between the initial model constructs and the final version.

Experiments to be performed have an effect on the design of model constructs. In both projects the model constructs have been adjusted, because they did not match the type of desired experiments, input parameters or performance indicators. The design of the domain specific simulation environment should take the experiments and their output more as a starting point.

5 REQUIREMENTS

The requirements described in this section are based on lessons learned from the case studies and laboratory experiments. The requirements are divided into four sets. The first two sets of requirements deal with the domain specific simulation environment and its individual model constructs. The last two regard the design and use of a domain specific simulation environment. In each of these requirements the starting point is working with professional persons that are trained in their domain. The requirements will help novices who try to develop domain specific simulation environments, but are mainly aimed to make professionals better.

The ‘requirements for domain specific simulation environments’ deal with what can be expected from a domain specific simulation environment in its use and its ability to instantiate the model construct to represent particular systems. The ‘requirements for model constructs’ helps to define the minimum characteristics of individual model constructs. Section 2 indicated that a model construct could be anything. A developer can still represent any system element, but these requirements will limit the freedom of developers of domain specific simulation environment in the way of representing and structuring. Matching the requirements for model constructs will increase the clarity and understanding for model developers that will use these model constructs.

The 'requirements for design approach for domain specific simulation environment' are generic requirements for design of a domain specific simulation environment. The design approach that follows from these requirements supports development of a domain specific simulation environment that matches the characteristics for individual model constructs and the domain specific simulation environment as a whole. Finally, 'requirements for guidelines for use of domain specific simulation environment' are requirements to enhance guidelines to support model developers in instantiating a simulation model from a domain specific simulation environment and use the model for experimentation with a system.

5.1 Requirements for Domain Specific Simulation Environments

Usable within several simulation studies; A domain specific simulation environment should be suitable for instantiating simulation models for several simulation studies. This means that the focus in the design should not be on only one case, like it happened in the design of the domain specific simulation environments of the AGVs case study.

Usable at different levels of reduction; Within different simulation studies, but also in the same simulation study, often the system is analyzed at different levels of detail. For example, simulation models of the airport Schiphol have been analyzed as a complete system and with a focus on only the check-in of KLM. This requires that within a domain specific simulation environment several model constructs are available that represent one system element at different levels of reduction by representing different details. Obviously the use of different levels of reduction can result in connectivity problems and issues in understanding of syntax (Reynolds et al, 1997). How these issues are triggered depends on the domain and types of problems in this domain.

Clearly define the ability of use and when not to use; The applicability of a domain specific simulation environment depends on the model constructs that are provided. In some situations the domain specific simulation environment is simply not suited for use for a certain problem. Adjusting the model constructs or develop new constructs might solve this issue.

Extendable with new model constructs; A domain specific simulation environment that contains representation of all the events that occur in a chosen problem domain can never be developed. Therefore, there should always be a possibility to add new model constructs to the domain specific simulation environment, however, the domain specific simulation environment should be configured in such a way that new model constructs can be added to represent system elements at a different level of reduction and not to include any new type of operations.

Adjust existing model constructs; like the domain specific simulation environment will change, it might also occur that existing model constructs need to be adjusted. This could include new functionalities in a model construct or adjustment of existing functionalities.

Support material to gain trust; Model developers showed in the laboratory experiments that they require support to gain trust in the domain specific simulation environment. Experts had detailed and technical questions, before they would be fully trusting a domain specific simulation environment. Novices showed that their information need is less specific. Support that helps both types of users in their model development and use should be available with the domain specific simulation environment. This support material should show the users of the domain specific simulation environment how system elements behave in the model constructs.

Additional analysis tools or instruments to support understanding of the outcomes of simulation models; Model constructs of domain specific simulation environment can provide a lot of statistics and model developers get lost in the enormous amount of data provided by each experiment. The AGV-case showed the added value of an additional tool and the airport case showed the need for such a tool. Any domain specific simulation environment should be provided with this tool to enable model developers to analyze and observe the outcome of their simulation model that represents their system effective.

Ability to compose simulation models that are understandable for problem owners; The simulation models that are composed should represent that what the problem owner expects to see. Therefore, the domain specific simulation environment should consist of sufficient model constructs that all can represent one or more system elements in a way understandable and recognizable for a problem owner.

Ability to compose simulation models that show valid behavior; Use of the model constructs of a domain specific simulation environment should provide a simulation model that represents the system in a valid way. This requires that all model constructs of the simulation model are tested and validated for their individual behavior and the combination of model constructs is also tested. The instantiation of model constructs could result in a simulation model that is an invalid system representation, but parameterizing of the model constructs should in the end result in a valid representation, which is acceptable by the model developer and problem owner.

5.2 Requirements for Model Constructs of Domain Specific Simulation Environments

Follow basic rules of systems thinking and software engineering; Within the domain of systems thinking and software engineers for years components have been designed and developed. The concepts for decomposition and design of interfaces are very important to follow within the development of model constructs of domain specific simulation environments (Gigch, 1991; Szyperski, 2001).

User interface for parameterization; Model developers need to be able to set parameters of the model construct without needing to dive into the model constructs that the domain expert used to define the specific model construct.

User interface for parameterization; Model developers need to be able to set parameters of the model construct without needing to dive into the model constructs that the domain expert used to define the specific model construct.

User interface in terminology of problem owner; The user interface of model constructs should be understandable by the model developer and by the problem owner. The terminology used in the model constructs, mainly in the user interfaces to the problem owner, should contain terms that the problem owner is used to.

Not too many functionalities in one model construct; In the case studies of the check-in counters at airports the model constructs kept on extending and growing with new performance indicators, parameters and functionalities to achieve a good representation of every check-in counter that was observed. These extensions made it hard to maintain the model construct by developers of domain specific simulation environments and hard for model developers to use the model construct (too many parameters to set).

Performance indicators that make sense to problem owners; The problem owner is often interested in other things than the default statistics that are generated by use of resources. The case study of the airport showed the need for reporting numbers of passengers that are too late for a flight. The model constructs should enable these kinds of performance indicators, because these will enable the problem owner to trust the model and its outcome, and to make better use of the outcomes without lots of calculations after completion of the simulation study.

Model constructs separated for physical and control system elements; Separating the controls from the infrastructure provides smaller model constructs and better capability to vary either control behavior or infrastructure composition. Model constructs that enable this separation turned out to be easier to use and more flexible.

Generate errors and warnings for model developers during model development; Model developers expect the domain specific simulation environment to support them in their model development. Important in support is guidance if a model developer is doing things that are not entitled or matching with the model constructs.

5.3 Requirements for Design Approach for Domain Specific Simulation Environment

Support developers of domain specific simulation environments; The design approach should help developers in the process, and not make it unnecessarily difficult.

Provide insight into the complexity to involved problem owners and future model developers; Developers of domain specific simulation environments cannot develop without input from a problem owner. Time and knowledge of a problem owner is required to be able to develop model constructs that can be used in several simulation studies and represent system elements valid and understandable.

Provide insight in required data/information/system knowledge; Problem owners should provide the developer with all kinds of knowledge. This amount of required domain knowledge is much more than just a short description of a particular sub-system.

Provide overview of deliverables; Problem owners and model developers need to be able to gain trust in the domain specific simulation environment, a first step is that they gain trust in the development process. Deliverables of the developer of the domain specific simulation environment will enable trust and understanding in the (design of the) domain specific simulation environment.

5.4 Requirements for Guidelines for Use of Domain Specific Simulation Environment

Make sure that all steps of a simulation study are performed; A simulation study using a domain specific simulation environment is slightly different from a simulation study using a generic simulation environment. Still all process steps of a full simulation study should be performed to carry out a valid simulation study.

Pay attention to trust of model developers in the domain specific simulation environment; Model developers who work for the first time using a domain specific simulation environment are forced to use a different concept than what they are used to. The laboratory experiments showed that trust needs to be gained in the capabilities of the domain specific simulation environment, before a model developer can perform a successful simulation study. Obviously this "brainwashing" of model developers should only be performed if the model constructs of the domain specific simulation environment match with the problem within the domain.

Pay attention to selection of model constructs; a domain specific simulation environment that contains several model constructs to represent the same system elements at different levels of reduction provides the model developer with difficulties. The guidelines should provide the model developer with insight how to overcome these difficulties.

6 DISCUSSION AND FUTURE RESEARCH

The defined requirements partly make clear why domain specific simulation environments are not popular among model developers. A lot of the requirements are in our opinion not matched by the existing domain specific simulation environments. We think that the domain specific simulation environments mainly lack the support for model developers to gain trust. For example the tanks and flow process template of Arena does not consists of model constructs like a blending vessel, or include important activities like cleaning, and therefore it is still not related to the system perspective of domain experts. Furthermore, the domain specific simulation environments are still reasonably generically applicable.

The requirements that have been defined in the previous section of this paper can be used to define a reference architecture for domain specific simulation environments. This generic structure will enable model developers to better understand how the model constructs of the domain specific simulation environment can be used. In addition the requirements enable to define minimum configurations of model constructs of these domain specific simulation environments. This definition of minimum configurations should enable that the design of the model constructs of the domain specific simulation environments can be adjusted and extended with alternative and new model constructs. These additions to the domain specific simulation environment are expected to be easy to understand by model developers, thanks to the requirements for a domain specific simulation environment, and the presented guidelines for design and use.

REFERENCES

Altiok, T; W. Xiong; M. Gunduc. 2001. A capacity planning tool for the Tuxedo middleware used in transaction processing systems. In: B.A. Peters; J.S. Smith; D.J. Medeiros; M.W. Rohrer (Eds.) *Proceedings of the 2001 Winter Simulation Conference*, 502-507. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Arends, D. 1999. *Using object-oriented simulation for a quantitative approach of the terminal concepts,* master thesis, Delft University of Technology, Netherlands

Balci, O. 1997. Principles of simulation model validation, verification, and testing. *Transactions of the Society for Computer Simulation International*, 14 (1): 3-12.

Banks, J.; F. Azadivar; D. Ferring; J.W. Fowler; D.W. Halpin; A.M. Law; M. Manivannan; W.S. Murphy. 2001. Panel session: the future of simulation. In: B.A. Peters; J.S. Smith; D.J. Medeiros; M.W. Rohrer (Eds.) *Proceedings of the 2001 Winter Simulation Conference*, 1453-1461. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Bapat, V.; D.T. Sturrock. 2003. The Arena product family: enterprise modeling solutions. In: S. Chick; P.J. Sanchez; D. Ferrin; D.J. Morrice (Eds.) *Proceedings of the 2003 Winter Simulation Conference,* 210-217. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Barton, R.R.; P.A. Fishwick; R.G. Sargent; J.O. Henriksen; J.M. Twomey. 2003. Panel: simulation – past, present and future. In: S. Chick; P.J. Sanchez; D. Ferrin; D.J. Morrice (Eds.) *Proceedings of the 2003 Winter Simulation Conference*, 2044-2050. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Davis, P.C.; P.A. Fishwick; C.M. Overstreet; C.D. Pegden. 2000. Model composability as a research investment. In: J.A. Joines; R.R. Barton; K. Kang; P.A. Fishwick (Eds.) *Proceedings of the 2000 Winter Simulation Conference*, 1585-1591. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Diamond, R.; J.O. Henriksen; C.D. Pegden; A.P. Walker; C.R. Harrell; W.B Nordgren; M.W. Rohrer; A.M. Law. 2002. The current and future status of simulation software (panel). In: E. Yücesan; C.H. Chen; J.L. Snowdon; J.M. Charnes (Eds.) *Proceedings of the 2002 Winter Simulation Conference*, 1633-1640. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Evers, J.J.M.; S.A.J. Koppers. 1996. Automated guided vehicle traffic control at a container terminal, *Transportation research part A-Policy and practice*, 30 (1): 21-34

Gigch, J.P. van. 1991. *System design modeling and metamodeling*, Plenum Press, New York.

Harrell, C.R.; R.N. Price. 2002. Simulation modelling using Promodel Technology. In: E.Yücesan; C.H. Chen; J.L. Snowdon; J.M. Charnes (Eds.) *Proceedings of the 2002 Winter Simulation Conference,* 192-198. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Heijden, M.C.van der; A. van Harten; M.J.R. Ebben; Y.A. Saanen; E.C. Valentin; A. Verbraeck. 2002. Using Simulation To Design an Automated Underground System for Transporting Freight Around Schiphol Airport, *Interfaces*, 32 (4): 1-19.

Kasputis S.; H.C. Ng. 2000. Composable simulations. In: J.A. Joines; R.R. Barton; K. Kang; P.A. Fishwick (Eds.) *Proceedings of the 2000 Winter Simulation Conference*, 1577-1584. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Page, E.H.; J.M. Opper. 1999. Observations on the complexity of composable simulation. In: P.A. Farrington; H.B. Nembhard; D.T. Sturrock; G.W. Evans (Eds.) *Proceedings of the 1999 Winter Simulation Conference*, 553-560. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Pater, A.J.G.; M.J.G. Teunisse. 1997. The use of a template-based methodology in the simulation of a new cargo track from Rotterdam harbor to Germany. In: S. Andradottir; K.J. Healy; D.H. Withers; B.L. Nelson (Eds.) *Proceedings of the 1997 Winter Simulation Conference*, 1176-1180. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Petty M.D.; E. W. Weisel; R. R. Mielke. 2003. Computational Complexity of Selecting Models for Composition" In: *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 517-525.

Reynolds, P.F.; A. Natrajan; S. Srinivasan. 1997. Consistency maintenance in multiresolution simulation. In: *ACM Transactions on Modeling and Computer Simulation*, 7 (3): 368-392.

Saanen, Y.A. 2004. *An approach for designing robotized marine container terminals*, Doctoral Dissertation, Delft University of Technology, Netherlands.

Sadowski, D.; M.R. Grabau. 2000. Tips for successful practice of simulation. In: J.A. Joines; R.R. Barton; K. Kang; P.A. Fishwick (Eds.) *Proceedings of the 2000 Winter Simulation Conference*, 69-76. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Sol, H.G. 1982. *Simulation in information systems development*, Doctoral Dissertation, Rijksuniversiteit Groningen, Netherlands.

Szyperski, C. 2001. *Component Software:Beyond Object-Oriented Programming.* Addison-Wesley, New York.

Valentin, E.C. 2003. Evaluation of Check-in Processes at JFK International Airport using Discrete Event Simulation. In: R. Clute (Ed.) *Proceedings of International Conference on Applied Business Research*, CD-ROM, 11 pages.

Valentin, E.C.; A. Verbraeck; H.G. Sol. 2003a. Effect of Simulation Building Blocks On Simulation Model Development. In: A. Ibarra (Ed.) *Proceedings of International Conference of Technology, Policy and Innovation*, 54-61.

Valentin, E.C.; A. Verbraeck; H.G. Sol. 2003b. Advantaged and disadvantages of building blocks in simulation studies: a laboratory experiment with simulation experts. In: A. Verbraeck, V. Hlupic (Eds.), *Proceedings ESS'2003 – 15th European Simulation Symposium,* 141-148.

Verbraeck, A.; Y.A. Saanen; E.C. Valentin. 1998. *Terminal rapportage Simulatie Experimenten-deel 1* (Dutch). Connekt, Delft.

Verbraeck, A.; E.C. Valentin. 2002. Simulation Building Blocks for Airport Terminal Modeling, In: E. Yücesan; C.-H. Chen; J.L. Snowdon; J.M. Charnes (Eds.) *Proceedings 2002 Winter Simulation Conference*, 1199-1206. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

EDWIN C. VALENTIN is a simulation consultant at Systems Navigator in The Netherlands, and a part time researcher in the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology. His specialty is the development of domain specific simulation environments for discrete event simulation. Edwin participates in the BETADE research program on developing new concepts for designing and using building blocks in software engineering, simulation, and organizational modeling.
His email address is:
edwin.valentin@systemsnavigator.com.

ALEXANDER VERBRAECK is chair of the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and a part-time full professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation for real-time control of complex transportation systems and for modeling business systems. His current research focus is on development of generic libraries of object oriented simulation building blocks in C++ and Java. His email and web addresses are:
a.verbraeck@tbm.tudelft.nl
www.tbm.tudelft.nl/webstaf/alexandv.