Aircraft integration real-time simulator Modeling with AADL for architecture tradeoffs.

Jean Casteres, Tovo Ramaherirariny
Simulation Industry Department
Airbus
316 Route de Bayonne F-31060
Toulouse Cedex 9, France
e-mail: Jean.Casteres@airbus.com

***Abstract*—In today's aircraft, system complexity increases are making it particularly challenging for engineers to validate systems architectures. To ease this burden, the integration test rig, often known as the "iron bird" integration simulator, has been developed, and allows testing of real systems in a simulated environment.**

The computing host platform and interface equipment used in the integration simulator, are evolving rapidly. The capability to predict the performance of both the simulation application and the infrastructure on which it runs, is crucial in order to select the proper architecture for the future test rigs.

This paper presents the results of an AADL development that simulates the test rig simulator in order to predict its needs. We illustrate the use of model based engineering techniques on a real industrial application where we simulate the simulator in order to architect its computing infrastructure.

Firstly, the simulation application model built with AADL language is presented. Secondly, the producer-consumer paradigm is introduced and it is shown how it is used to model the simulation infrastructure host platform. Thirdly, the time reference used to abstract time in the simulation is presented. And finally the capacity of the AADL simulation to match the simulators currently used in our company is illustrated.

Keywords: modeling, real-time, simulation, AADL.

I. INTRODUCTION

TODAY'S aircraft system complexity increases at a fairly high pace. Demanding requirements from the airline companies trigger complex solutions from the aircraft manufacturers to reduce fuel consumption, increase capacity and reduce noise to name a few. These new complex systems make it particularly challenging for engineers to select between aircraft architecture alternatives.
In parallel, to manage this complexity, computer systems are broadly used in the specific technology domains involved in the making of the aircraft product.

Simulation is often a way to reduce the costs of testing aircraft complex systems, and chains of simulation platforms, ranging from the research simulator to the integration test rig, support the aircraft program.

This paper presents the results of the simulation of the integration test rig using the ***Architecture Analysis and Design Language*** (AADL). The integration test rig is a simulator used in the development chain of the aircraft program to test real aircraft equipment as they become available. The project of main host computer renovation identified that being able to predict both the needs of the next integration test rig and the capabilities of the infrastructure was a necessity. Modeling the simulator and abstracting the simulation application from the hardware and the software it runs on, will allow the definition of the architecture of the simulator based on the application it is simulating [1]. This paper presents the follow up on this idea and the capabilities of the environment developed.

In a first part, the way the simulation application of the aircraft was modeled in AADL is presented. The selection of the AADL environment [2] as well as the key parameters used, are highlighted.

In a second part, the paradigm of producer consumer is introduced. The method used to model the hardware using this paradigm is then described.

In a third part, the model for time handling in the environment is presented. The fundamental differences between three distinct usages of time in the simulation, and their impact on the test rig simulator environment is depicted.

Finally, results of the validation campaign are presented, the measures probed on the real test rig are compared with the numbers predicted with the AADL environment.

II. MODEL OF THE SIMULATION APPLICATION

A. Presentation of the Test rig integration simulator

Aerodynamics, engines, flight dynamics, electric and hydraulic systems are all knowledge domains required to assemble an aircraft. Different simulation platforms have been built to help design and production throughout the aircraft program cycle. These simulators cover the research simulator (EPOPEE), the development simulator (A/C-1 and desktop simulators) and the integration simulator A/C-0. These simulators support the aircraft development from 5 years, 2.5 years and one year before the first flight respectively.

The iron bird integration concept dates back to the

978-3-9810801-5-5/DATE09 © 2009 EDAA

Concorde program: compatibility checking between various aircraft systems can be performed at a lower cost on the ground [3]. The compatibility between real aircraft systems is verify and validated on the integration simulator test rig: the simulator shall perform simulation in real time since it needs to stimulate real avionics equipments and interface with a real pilot.

The integration simulator connected to the iron bird, or aircraft zero (A/C-0) simulator, is used to prepare the first flight but also to participate in aircraft certification, and remains operational throughout the aircraft program lifecycle.

The integration simulator development process provides the ability to model the natural flight loop, the engines, the aircraft environment linked with the real aircraft equipment as they come in.

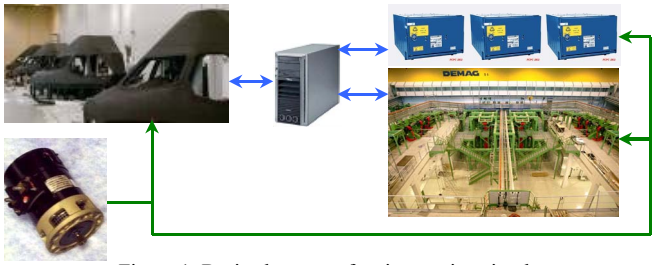

Figure 1: Basic elements of an integration simulator

The systems required to assemble an integration simulator are following a rapid pace of evolution and change. The changes come from the aircraft needs. An aircraft program manages the various domains of expertise that need different simulation results. Various test rigs can be geographically separated.

New technologies that the program would like to target need to be introduced and are first tested on the integration simulator. The underlying technology of the host computer and the integration interfaces require upgrades and are rapidly changing, which is a trend that will likely accelerate in the future with such features as instruction level parallelism and explicitly parallel instruction computers, requiring compiling tool support [4].

The objective of our simulation is to be capable of modeling the integration simulator in this context.

The software application as well as the hardware architecture on which it runs, are the two main bricks that need to be modeled in a common environment in order to examine the performance tradeoffs that can be achieved by trying different architectures or technological solutions.

B. Selection of the simulation environment

A preliminary study [5], has highlighted a wide offering in rapid prototyping environments.

Because of the real-time capability displayed by the integration simulator, the selected environment had to be capable of scheduling the various processes that would need to be modeled. This capability allows mimicking the behavior of the integration simulator architecture in a purely simulated environment. Both the hardware and the application software needed to be swiftly modeled with the same environment in order to be able to become a rapid prototyping environment.

The UML language, the scheduling performance and time profile from [6], open and commercial solutions such as I-Logic Rhapsody, Rational Rose and RealTime have been looked at and analyzed to compare services and performance. Also, integrated environments such as Ptolemy, Workbench, Csim, Matlab and Scilab were also scrutinized at the time.

None of these could exactly match all the needs and constraints. Finally, the AADL was selected because of its ability to model complex software and hardware architectures, its original design for simulation [2], and the ease with which new properties can be added to objects. The availability of a scheduling tool such as Cheddar [7] was also a decisive advantage.

C. Key parameters for the application software model

The environment needs to meet certain constraints in order to be able to simulate the test rig simulator:

- Model the aircraft simulation application
- Model the host computer and the interfaces
- Collect performance information on the processing power requirements, the IO activity, the buses and network traffic
- Display results in a reliable format to facilitate architecture tradeoffs with a sufficient accuracy level
- Provide a convenient interface to change the performance parameters in order to perform tradeoffs
- Easily connect to other simulation frameworks

Two main tradeoffs need to be supported: computing power balancing and distribution of the simulation across distant nodes.

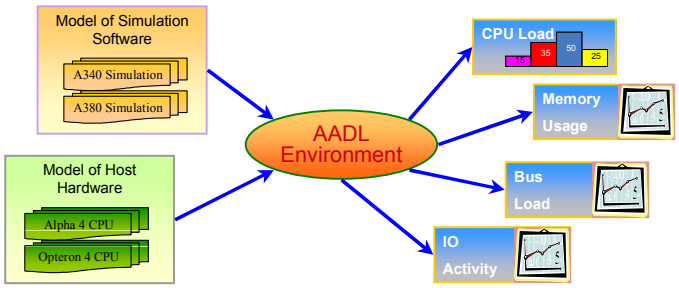

Figure 2: Basic elements of the 'integration simulator's' simulator

The key parameters collected to evaluate various architectures would be computing power, IO activity, memory usage, and bus transaction loading. A software model of the various tasks of the integration simulator software application had to be put in place on top of a hardware model capable of mimicking the processors, the crossbars, the buses to the interface cards and the network cards.

The overall system was complex and a common concept was needed to help represent all the elements of the integration simulator architecture.

III. PRODUCER-CONSUMER PARADIGM

The goal is to model the flow of execution of the simulator.

This means, all the software tasks executing on the architecture with the ability to compute the central processing unit (CPU) usage, memory consumption and IO traffic together with their respective latencies. The performance of the system and its bottlenecks are the point of observation that are needed for the rapid prototyping environment. To achieve this goal, a common paradigm is introduced.

A. Simulation of the simulator test rig

Using the transactions, and the consumer producer paradigm explained hereafter, the hardware infrastructure and the simulation application will be assembled in the overall AADL environment to simulate the test rig.

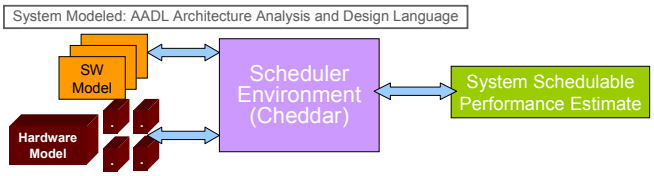

Figure 3: AADL environment used to simulate the integration simulator

The aircraft simulation application uses a modeling language, such as AADL, to model :

- The Simulation application software structure
- The Simulation application hardware platform and infrastructure

B. Producer Consumer Basic paradigm

The process is the executing instance of a program [8] (the word "task" is used as a synonym). A task has a unique identification number and a priority is associated to that number. The scheduling policy algorithm computes process priority as explained in [9].

An authority, the scheduler, arbitrates amongst a set of requests to select that one that should be active according to the scheduling policy. The active one will be granted the resource it has requested.

The consumer of a resource issues the requests to the central referee: the scheduler, that can be viewed as a producer of the resource.

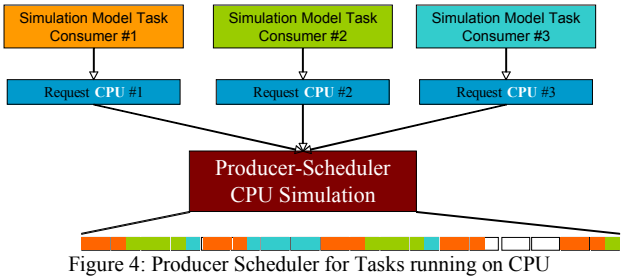

Figure 4: Producer Scheduler for Tasks running on CPU

In the computer world, the task is a process that wishes to 'run' and the machinery resource to enable this, is the CPU. The request of the tasks for the CPU computing time can be viewed as requests to a scheduler, which grants the CPU machinery resource.

The simulation application of the test rig integration simulator is modeled by tasks. Tasks consume the CPU resource: a task represents a request for CPU usage as in Fig.4. The scheduler schedules the task and computes the load of the system.

The modeled software task is not only requesting computing time but also other resources such as shared memory accesses and input/output network accesses. A software simulation application task that is modeled in our environment is an entity that can request three different types of services: processing power, memory and IO accesses.

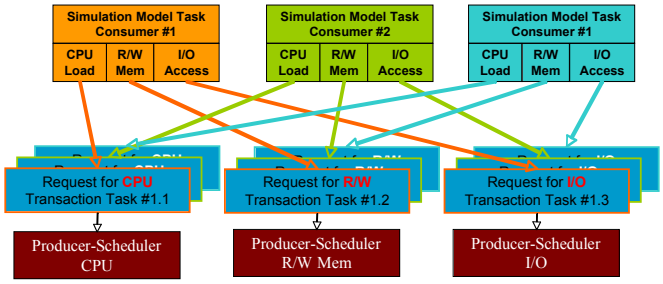

Figure 5: Resources requested by a modeled software task

The producer of a resource does not produce the total amount of the resource instantaneously. Requests for the resources are modeled as transactions and a scheduler is used to represent the fact that requesting tasks or transactions receive the resource in sequence.

In addition to the CPU requests for processing power, Read/Write requests to the dataflow (i.e.: a shared memory segment) can be converted into transaction tasks as well. Similarly, I/O requests to and from the dataflow are converted to transaction tasks.

Transaction tasks are scheduled by the producer-scheduler in the same fashion for all three types of resources. So the producer-scheduler also represents the bus usage: as an example hyper-transport or PCI.

The scheme is real-time independent: it follows the simulation tick sequence that reflects the sequence in which requests are answered. Time stays a variable that can be manipulated by the simulation in order to compute the penalty of the various latencies encountered.

The same paradigm can also be used to model outer rings of the communication such as Internet.

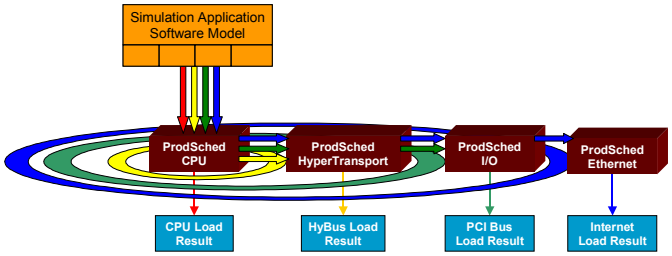

Figure 6: Hardware infrastructure modeling paradigm

Therefore, the infrastructure hardware has been modeled as three concentric circles providing transaction based services to the requesting simulation model tasks. The different elements making a computer infrastructure have been assimilated to producers and consumers of resources, because these elements can handle concurrent requests and arbitrate them. This is because their behavior has been identified as similar to that of

a scheduler: the entire environment can be modeled using the paradigm.

C. *Transaction description*

The simulation model task consumes resource services provided by the producers; each producer is a scheduler that provides the resource when answering a transaction request. A transaction is the request/response sequence that mimics the request-and-grant of the resource in the "real" world, the performance penalty to be paid is updated in a variable "carried with" the transaction itself.

For example the transaction for a shared memory access can be represented on the Fig.7 below.

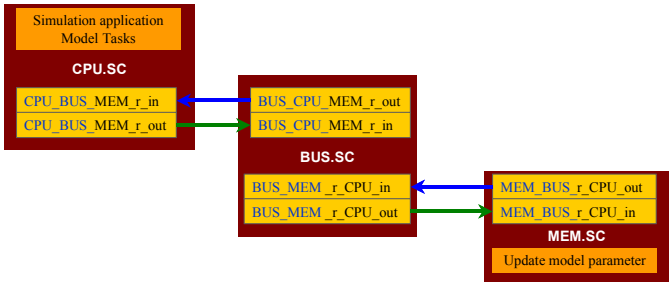

Figure 7: Transaction chain

The transaction is modeled using two tasks with a precedence link. The parent task becomes active and because it is linked to the child task, the child will become active once the task capacity is exhausted. The child tasks can be on a different scheduler, hence on another resource producer.

Using the precedence link allows two schedulers to represent a transaction flowing and being granted a resource. It takes one simulation tick for a transaction to flow from one producer to the next (one scheduler to the next). This simulation tick is not related to the simulated time: the last model in the transaction chain only updates the variable property 'time' with the simulated value that represents the time the transaction shall have consumed in real world.

The tasks used in the transaction and the producer/scheduler involved are summarized in the table below.

Scheduler Origine	Tâche origine	Tâche liée	Scheduler Destination
CPU	CPU_BUS_MEM_**r_out**	BUS_CPU_**MEM_r_in**	BUS
BUS	BUS_MEM_**r_CPU_out**	MEM_BUS_**r_CPU_in**	MEM
MEM	MEM_BUS_**r_CPU_out**	BUS_MEM_**r_CPU_in**	BUS
BUS	BUS1_P1_**MEM_r_out**	P1_BUS1_**MEM_r_out**	CPU

Table 1: Transaction Tasks modeling

The proposed environment makes a dual use of the tasks provided in the AADL language:

- Modeling of the tasks of the software application
- Modeling of the transactions between the producers (schedulers) of various resources; the capacity of the task representing the number of transactions

The concepts of abstraction used to model the software application and the hardware infrastructure have been presented. This highlighted the reasons why AADL and its simulation environment was the best solution available for the targeted rapid prototyping environment.

IV. TIME REFERENCE ABSTRACTIONS IN REAL-TIME SIMULATIONS

The test rig encompasses real aircraft equipment that has its own source of time reference. It is the reason why real-time constraint applies to the test rig simulation. But this is not the case for the rapid prototyping environment whose goal is to study tradeoffs based on performance prediction.

In order to estimate the performance of the foreseen test rig infrastructure, time is a physical variable that needs to be taken into account and computed to figure if the real-time deadline can be met for the system. In this second order simulation, time is therefore a result of the model. It is a physical dimension that is handled like any other (i.e.: length or a pressure), and will be computed relative to the other parameters belonging to the same reference.

To make the difference between what is computed and simulated from the environment itself a classification of time is proposed. It is not intended to be formal description of time as in [10] but rather a simplification.

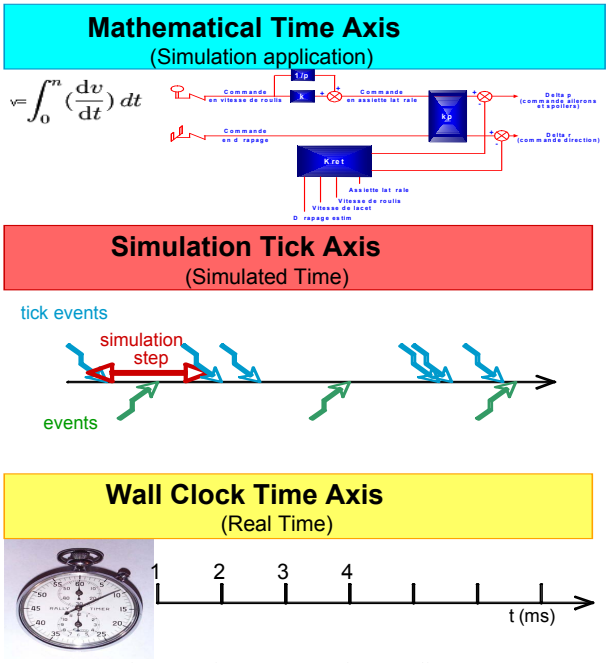

Figure 8: Time representation paradigm.

The mathematical time axis is the time value going into mathematical computation inside the models. The variable holds the time that is used by the models to compute equations using time as a parameter.

The simulation tick axis is the heart beat synchronizing the event steps between the models. The simulation kernel handles ticks between the simulation objects. A simulation step is the atomic synchronization of the simulator between two simulation ticks. Relative to the simulation tick time, the simulation steps do not necessary have the same length. Simulation objects can be "ticked" or not, in the later case the object is said to be purely event based.

Finally, the wall clock axis is real-time, that is, the physical time that naturally elapses. This axis is checked against the

mathematical and the heart beat axis, to make sure deadlines are met in order to provide real-time behavior of the simulator if it is required.

The objective of the integration simulator's simulation is to compute the performance and therefore to provide a value of the predicted elapsed time. Time is therefore on the mathematical axis and is a result of the penalty of the various transactions requested by the software or hardware models.

The producer/consumer paradigm is used to model the systems and the transactions will carry the value of the mathematical simulated time. For performance computations, the various resource producers update these values, according to the penalty or latency of the system.

The time concept and the consumer/producer paradigm presented above have been used to represent the integration simulator architecture using the AADL language.

V. VALIDATION AND RESULTS

In order to be able to use the AADL environment as a extrapolation tool that would allow the selection of future architectures, verification and validation has to be conducted.

The two main elements of the environment are the models of the integration simulator software and the models of the hardware infrastructure. The 'real' simulation application can be run on the integration simulator and performance numbers can be measured. The validation method is to compare the measurements on the real integration simulation platform and the predictions done using the AADL environment. The goal is to compare the accuracy of the developed model for the software application on one hand, and to compare the accuracy of the hardware infrastructure model on the other hand.

Accuracy, in this context, is the gap between the performance numbers measured on the real integration simulation platform and the performance computed with the AADL simulation environment.

A. First validation campaign of the processing load.

In a first step, a set of tests was conducted to verify the correlation in the performance estimations of the CPU load.

CPU load %	Set1	Set2	Set3	Set4	Set5	Set6	Set7	Set8
Platform	18	20	26	15	23	32	50	52
AADL SimEnv	22	18	28	19	24	33	42	57
Delta	22	-10	8	27	4	3	-16	10

Table 2: CPU load estimates

A 'set' represents a group of models of the simulation application software running on a unique CPU. On this first set, the results for which the gap were the largest, were identified as due to an under-estimate of the IO traffic in the models. However, the first validation campaign results encouraged the development of the environment.

It is to be noted, that a test on two hyper transport architecture alternatives was also conducted. The first architecture favored of inter-processor communications: two of the three hyper transport links were connected to another core on an eight-core architecture. The second displayed one of the links connected to an IO bridge. The AADL environment allowed to highlight that all software simulations not connected to interfaces would take benefit from the first architecture.

B. Application verification

Two steps are taken to validate the accuracy: one for the model of the hardware and one for the model of the application software.

First, the environment is used to validate new software architecture alternatives: it is important to make sure that on a given infrastructure, the environment can accurately simulate software architecture alternatives.

Second, the environment is used to study different hardware infrastructure alternatives for the integration simulator. It has been foreseen that the AADL environment should connect to other additional simulation environments capable of simulating specific elements of the infrastructure.

In order to verify the software model accuracy, the hardware infrastructure is fixed and a set of different application software is selected. The application software is modeled in the AADL environment and runs on the integration simulator platform while being measured.

		Set1	Set2	Set3	Avg.Delta
Application #1	Platform	19	17	30	
	AADL SimEnv	16	15	33	
	Delta	3	2	3	2,67
Application #2	Platform	21	50	52	
	AADL SimEnv	15	44	59	
	Delta	6	6	7	6,33
Application #3	Platform	15	30	40	
	AADL SimEnv	11	34	39	
	Delta	4	4	1	3,00

Table 3: Software application model validation

The sets represent a group of models of the simulation software running on a unique CPU. Since the load is a percentage of the CPU utilization, the delta is the difference between the load measured on the platform and the predicted load on the scheduled AADL environment.

The accuracy of the AADL environment prediction is between 2.7 and 6.4 percent of the measured value on the platform. This is a number that yields enough confidence in the AADL environment to be used as rapid prototyping environment to study application software architecture tradeoffs.

In order to verify the hardware model accuracy, the same software simulation application is run on models of different hardware infrastructures. The hardware infrastructure is modeled in the AADL environment and runs the integration platform software model. The integration platform runs the 'real' application whilst it is measured.

		Set1	Set2	Set3	Avg.Delta
Infrastructure #1	Platform	19	17	30	
Conf1	AADL SimEnv	16	15	32	
	Delta	3	2	2	2,33
Infrastructure #1	Platform	24	15	22	
Conf2	AADL SimEnv	21	12	25	
	Delta	3	3	3	3,00
Infrastructure #1	Platform	22	70	30	
Conf3	AADL SimEnv	18	100	34	
	Delta	4	30	4	12,67
Infrastructure #1	Platform	13	30	40	
Conf4	AADL SimEnv	11	34	39	
	Delta	2	4	1	2,33
Infrastructure #2	Platform	20	20	45	
Conf1	AADL SimEnv	18	17	41	
	Delta	2	3	4	3,00
Infrastructure #2	Platform	30	80	40	
Conf2	AADL SimEnv	25	100	37	
	Delta	5	20	3	9,33
Infrastructure #2	Platform	20	37	40	
Conf3	AADL SimEnv	19	39	39	
	Delta	1	2	1	1,33

Table 4: Hardware infrastructure model validation

The sets represent a group of models of the AADL simulation environment allocated to a unique CPU. Since the load is a percentage of the CPU utilization, the delta is the difference between the load measured on the platform and the predicted load on the scheduled AADL environment in the same manner as previously.

The accuracy of the AADL environment prediction is between 2.4 and 12.7 percent of the measured value on the platform. This is a number that yields enough confidence in the AADL environment for it to be used as a rapid prototyping environment to study integration simulator infrastructure tradeoffs.

The numbers displayed on the above tables are in a range that put the developed AADL environment in a very good accuracy range for a rapid prototyping environment. A small variation of the results can be observed that has two main root causes: small variation of the performance measurements used to feed the parameters of the AADL objects. Secondly, the IO traffic of some of the software models is a parameter of the AADL object. This parameter can be measured but had to be estimated for certain models.

An additional lesson learned was between the initial and final validation steps. Tools used to measure the application on the real platform were changed in order to have more accurate numbers. The introduction of a new Linux based host allowed more precise performance measurements tools that were in turn used in the software models and yielded better accuracy from the AADL environment predictions.

VI. CONCLUSION

The environment used allows predicting the infrastructure needed for an integration simulator, with an error bounded in the 5% to 15% interval.

Simulation on the domain of complex system is and will be more and more used. The models used will reflect reality with a higher degree of fidelity, therefore requiring more performance from the simulation infrastructure.

Convergence in the simulation world is taking place already: commonalities have been found in various aircraft domain simulations and other disciplines, laying grounds for even more complex systems of systems simulations.

Exploration of these new domains will bring new challenges on the modeling techniques: the simulation frameworks will need to provide better performing services, and the hardware infrastructure will need to evolve. This is also true for the input/output architecture with the real equipment whilst using low cost mainstream solutions.

These three points highlight the need for rapid prototyping environments such as the one presented above.

In addition, new virtualization techniques will eventually help all industrial parties participate in the building of the system; putting more stress on the simulation framework to accommodate all components of the simulation.

These new challenges will maintain the simulation discipline as a passionate field to be in. It highlights the need for formalisms to be standardized in that area. The AADL environment can be a good candidate for prediction of the systems of systems simulation infrastructures. The results presented in this paper show the precision and confidence that can be achieved in simulating the simulator for the complexity targeted in today's projects, and contributes to the need for a standardization initiative.

REFERENCES

[1] JM Calluaud, J Casteres, S Gaudaire, "Technology evolution of aircraft simulator for real equipments validation", Erts 2008, p5, 2008

[2] As-2 Embedded Computing Systems Committee, "SAE Architecture Analysis and Design Language (AADL)", AS5506/1, June 2006 see also: AADL home and Telecom Paris website.

[3] C. Coureau, D. Liot, B. Mattos, « Airbus Engineering Simulation Methods – Past and Future Trends », p5, §5, 2004

[4] John L. Hennessy, David A. Patterson, "Computing Architecture: A quantitative Approach", Third Edition, Morgan Kaufmann, p376, 2003

[5] G. Laurens, "Prototypage rapide d'un Simulateur et étude des performances", Internship Airbus, p18, 2006

[6] The OMG, "A UML Profile for MARTE", Modeling and Analysis of Real-time and Embedded Systems (MARTE), OMG doc. #: ptc07-08-04

[7] Cheddar : a Flexible Real Time Scheduling Framework. F. Singhoff, J. Legrand, L. Nana, L. Marcé. ACM SIGAda Ada Letters, volume 24, ACM Press, New York, USA. December 2004, ISSN:1094-3641. F. Singhoff and A. Plantec. "What is Cheddar", at AADL scheduling home site.

[8] W. Richard Stevens, "Advanced Programming in The Unix Environment", Addison-Wesley, 1993

[9] Giorgio, C. Buttazzo, "Hard Real-Time Computing Sysyems: Predictable Scheduling Algorithms and Applications", Kluwer Academic Publisher, 1997

[10] Bernard P. Zeigler, Herbet Praehofer, Tag Gon Kim "Theory of Modeling and Simulation", 2nd Edition, Academic Press Elsevier 2000

[11] Gerard Fleury, P. Lacomme, A. Tanguy "Simulation à Evènement discrets", Eyrolles, 2007