Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 229 – 236

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

A Modeling and Verification Approach to the Design of Distributed IMA Architectures using TTEthernet

Tiyam Robati[a], Abdelouahed Gherbi[a], John Mullins[b]

[a]*Dept. of Software and IT Engineering, École de Technologie Supérieure, Canada*
[b]*Dept. of Computer and Software Engineering, École polytechnique de Montréal, Canada*

Abstract

Integrated Modular Avionics (IMA) architectures complemented with Time-Triggered Ethernet (TTEthernet) provides a strong platform to support the design and deployment of distributed avionic software systems. The complexity of the design and continuous integration of such systems can be managed using a model-based methodology. In this paper, we build on top of our extension of the AADL modeling language to model TTEthernet-based distributed systems and leverage model transformations to enable undertaking the verification of the system models produced with this methodology. In particular, we propose to transform the system models to a model suitable for a simulation with DEVS. We illustrate the proposed approach using an example of a navigation and guidance system and we use this example to show the verification of the *contention-freedom* property of TTEthernet schedule.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Integrated Modular Avionics; TTEthernet; Model transformation; Verification; DEVS; Simulation

1. Introduction

Avionic systems belong to the class of safety-critical systems which have strict safety, reliability and real-time requirements. In contrast with the federated architectures where each software function is designed and deployed to use exclusive recourse, Avionic systems are now based on the Integrated Modular Avionics (IMA) architectures where several system functions, having different levels of safety and performance requirements, might be deployed on the same computing module[14]. IMA-based avionic systems are therefore mixed-criticality and require solid isolation and partitioning. These features are supported in IMA using operating systems and executives compliant with the ARINC 653 standard[2].

The IMA architecture supports distributed systems where the modules hosting the system functions are interconnected through a communication infrastructure that should also meet the same level of safety and timing require-

* Corresponding author. Tel.: +1-514 396-8465 ; fax: +1-514 396-8405.
E-mail address: Tiyam.robati.1@ens.etsmtl.ca, Abdelouahed.gherbi@etsmtl.ca, john.mullins@polymtl.ca

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.120

ments. Time-Triggered Ethernet (TTEthernet) is a standard extension of Ethernet [11] enhancing the predictability and determinism of Ethernet to meet strict real-time and safety requirements. To this end, TTEthernet is based on the time-triggered communication paradigm [13] and therefore establishes a system-wide time base implemented through a synchronization of the clocks of the end systems and switches. This results in bounded latency and low jitter. TTEthernet offers fault isolation mechanisms to manage channel and nodes failures and integrates three data flow: Time-Triggered (TT) data flow which is the higher priority traffic; Rate Constrained (RC) traffic, which is equivalent to AFDX traffic, and Best Effort (BE) traffic.

Therefore, IMA architectures interconnected with TTEthernet provides a platform to deploy avionic systems and applications with the required features to meet their requirements. In this platform, the error isolation is provided both at the level of the modules through the (time and space) partitioning and at the level of the network integrating differentiated data traffics.

In this paper, we focus on the verification of avionic systems deployed on the platform characterized earlier. Such systems are however complex and their design is challenging. In order to manage the complexity of such systems, we proposed in our previous work [4], an extension to the AADL modeling language to support the modeling of distributed systems composed of IMA modules with TTEthernet as a communication infrastructure. In this paper, we leverage this extension in order to further support the verification of the AADL models with respect to TTEthernet main scheduling properties and constraints. We propose to perform the verification step using a simulation-based approach [1,6], which will check the input system model with the scheduling properties of TTEthernet specifications. We use DEVS simulation environment to undertake the simulation process.

This remaining part of this paper is organized as follows: We present the details of our proposed approach in Section 2. We describe in Section 3 the application of the proposed approach with an illustrative example. In Section 4, we succinctly review the most close related research works to ours. Finally, we conclude the paper and outline our future research work in Section 5.

2. Proposed Approach

The overall architecture of our approach to model and verify distributed IMA systems using TTEthernet as communication infrastructure is depicted in Figure 1. The source system model is an AADL model, which represents an avionic application deployed on a distributed IMA system interconnected using TTEthernet. This model is actually an instance of AADL-TTEthernet metamodel presented in [4], which extends the AADL core metamodel to support the modeling of IMA architectures interconnected with TTEthernet.

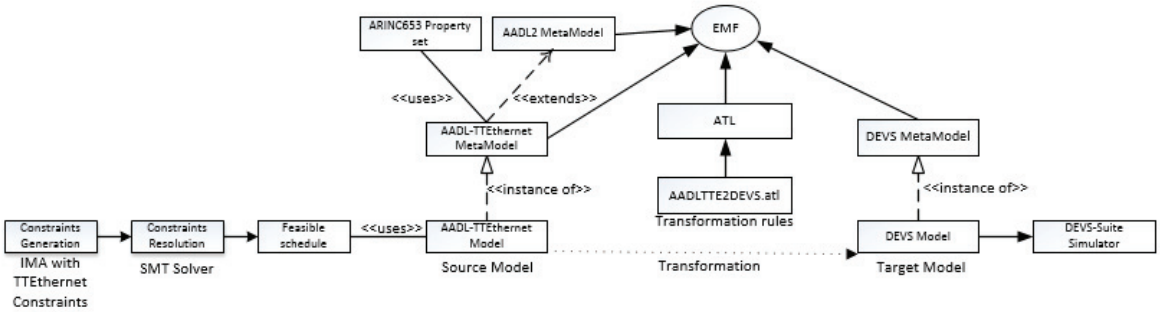

Fig. 1. Overall Architecture

The AADL-TTEthernet metamodel captures the main concepts and characteristics of the SAE TTEthernet standard AS6802 [11]. It describes also the structure of IMA architecture, which maps to the ARINC 653 property set defined in the standard AADL annex [2]. Therefore an instance of the AADL-TTEthernet metamodel is a model of a particular distributed avionic system deployed on a IMA architecture interconnected with TTEthernet. In addition, as shown in Figure 1, the instance model captures the system schedule, which is produced and incrementally integrated using our approach described in [12].

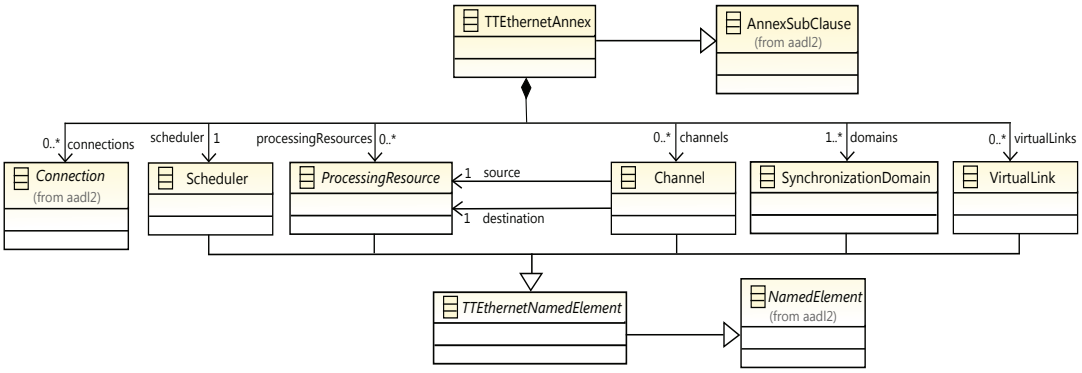

Fig. 2. Overview of AADL-TTEthernet metamodel

The AADL-TTEthernet metamodel is partially illustrated in Figure 2. In particular, the *Processing Resources* abstraction represents active hardware components of the network. All processing resources have *features* including parameters, access to physical buses or ports (i.e. interfaces for frames inputs and outputs). *Processing Resources* are divided into *Networking Resources* like *Switches* or *Computing Resources* like *Modules*. Partition is a group of time slices in a major frame (MAF) on a module. According to ARINC 653 standard[2], each function executes periodically within a partition where it is isolated from all other sharing core modules. Frame is a unit of transmission, a data packet of fixed or variable length, encoded for digital transmission over a communication link. A frame could be divided to *Protocol Control Frame* (PCF), *Time Triggered (TT)* frame, *Rate Constraint (RC)* frame or *Best Effort (BE)* frame. Finally, a cluster contains of several modules and switches that communicate together through links. A module contains one or several partitions that execute the overall functionality of module. Frames are input data of modules respectably partitions. Finally virtual links are the communication links defined by AFDX[3].

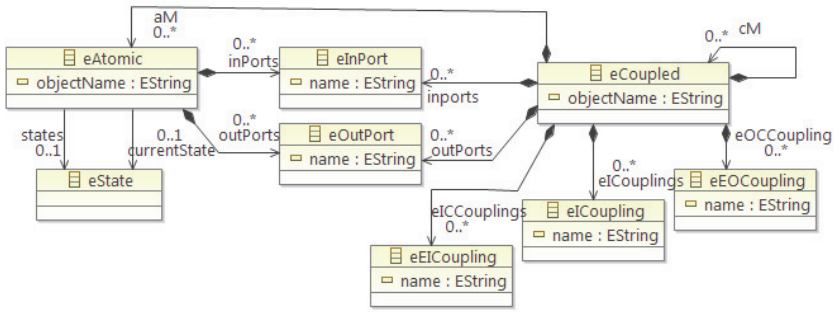

Fig. 3. DEVS metamodel[1]

In order to enable the simulation of the input system model, our approach is a classical M2M model transformation that is now a well-established pattern according to the model-driven engineering[21,22]. The target metamodel of this model transformation is a DEVS metamodel. We reuse a simplified version of this metamodel[1], which shown in Figure 3. In this metamodel, *Atomic* and *Coupled* are two levels that DEVS formalism provides for the system behavior description. At the lowest level, an *eAtomic* DEVS describes the autonomous behavior of a system as the Finite State automata. As well, it describes the way in which the *eAtomic* reacts to external inputs in order to generate the outputs. At the higher level, a *eCoupled* DEVS describes a system as a network of coupled components. For the latter, *eCoupled* DEVS reports how components influence each other and how the output of a component can become input of another one. *eInput* and *eOutput* of DEVS metamodel are assigned to model input and output of the system. Other classes of DEVS metamodel are used to model different possible situations when two *eCoupled* or two *eAtomic* or one *eAtomic* and one *eCoupled*, should be combined in order to represent the entire of the system.

The transformation of an instance of AADL-TTEthernet metamodel (i.e the system model) into an instance of the DEVS metamodel (i.e the simulaiton model) is achieved using a set of transformation rules specified using the the ATL model transformation language[23]. These model transformation rules are based on the general mapping shown in Table 1. The Partition and Switch in the source metamodel are two entities representing the behavior of system. Therefore, they can be mapped into *eAtomic* class of DEVS metamodel. A Module is mapped to *eCoupled* class in order to connect the partitions it includes. A Cluster, which regroups modules and switches, is mapped to *eCoupled*. A frame is input data of module and partition is mapped to *eInport* of DEVS. Virtual link is responsible of coupling module respectably partitions and switches, thus it is mapped to *eCoupled* of DEVS metamodel. For instance, the ATL transformation rules given in Figure 4 specify how the concepts module and partition in the source metamodel are transformed into the corresponding entities in target metamodel. The target model, generated by ATL transformation, is an intermediate model that can be used in the future to perform directly the model simulation realized by DEVS simulation environment.

```
rule Module2eCoupled {

   from a: aadltte!Module

   to AtomModule: DEVS!eCoupled(
       name<-a.name,
       inPorts<-thisModule.getFrames(a),
       aM<-a.partitions)}
```

```
rule Partition2eAtomic {

   from b: aadltte!Partition

   to AtomPartition: DEVS!eAtomic(
       name<-b.name,
       inPorts<-b.frames)}
```

Fig. 4. ATL transformation rules

Table 1. Mapping source model into target model.

Source Model	Target Model
Cluster	eCoupled
Module	eCoupled
Partition	eAtomic
Frame	eInport
Switch	eAtomic
Virtual link	eCoupled

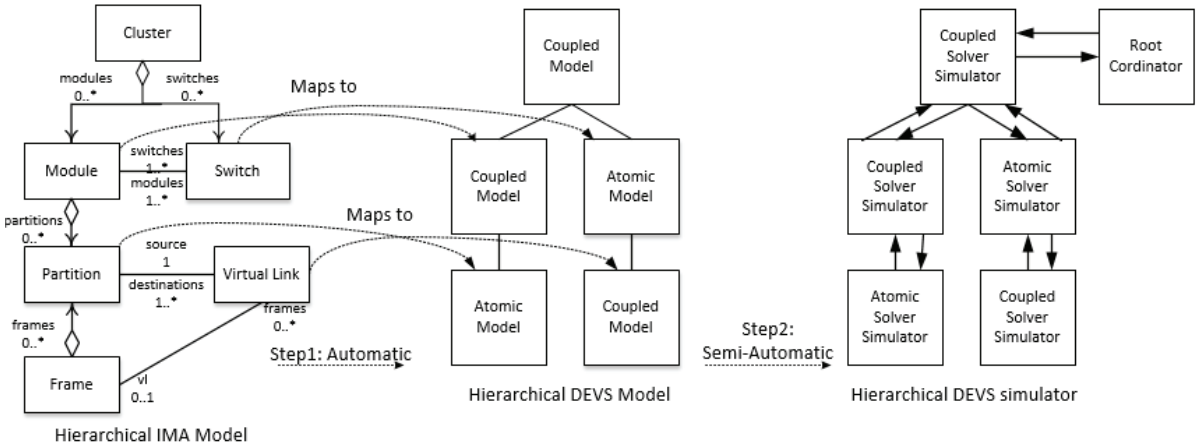

Fig. 5. Mapping a hierarchical model onto a hierarchical simulator

The instance model that results from the transformation step need to be refined in order to obtain a model fully suitable for simulation in a DEVS simulation environment. This refinement consists essentially in add the behaviors of source model into its implementation. To do so, we generate the java code corresponding to target model using of Acceleo[8], which is implementation of the Model to Text Language (MTL) standard. The behavior of source model is added to JAVA code obtained with Acceleo.

The main challenge to provide a model suitable for simulation with a hierarchical DEVS simulator is determining the sequence of DEVS activation at run time. More specifically, this challenge addresses the sequence of atomic or coupled in the entire simulatable model, and is tackled them by means of a *message-passing mechanism* of DEVS formalism[7]. The hierarchical DEVS simulator consists of *DEVS simulator*, *DEVS coordinator* and *message-passing mechanism* as demonstrated in Figure 5. The *message-passing mechanism* shown in Figure 5 by two direction arrows includes four categories of messages: an initialization method, an internal state transition message, an output message, and finally an input message to coordinator. This helps with controlling and monitoring the sequence of actions taken during the simulation. As illustrated in Figure 5, the mapping of a hierarchical IMA model into a hierarchical DEVS simulator is accomplished during step 1 and step 2. The hierarchical IMA model in this figure, represents an IMA architecture interconnected with TTEthernet. This model is mapped to the hierarchical DEVS model in step one. The hierarchical DEVS model is a DEVS model resulted in accordance with the mapping rules. In step 2, hierarchical DEVS model is mapped to hierarchical DEVS simulator which is the simulatable model.

3. Simulation of the Navigation & Guidance System

In this section, we present a case study to illustrate our proposed proposed approach. We present the system, how is it modeled using the AADL extension we proposed, the transformation process of this model into a DEVS simulation model, and finally the illustration of the verification of the contention free property using the simulation model.

3.1. System Description

In this case study, we consider a simplified navigation and guidance system[9]. As shown in Figure 6, the system is composed of four modules and two switches. The *Autopilot* (AP) module elaborates flight command to reach an altitude defined by the next way-point of the flight plan. The *Multifunction Control Display Unit* (*MCDU*) presents an interface between the system and the crew. The *Flight Management* (*FM*) sends periodically the next way-point*(pos)* to *AP*. The *Flight Warning* (*FW*) reports the equipment status *(sens-stat)* to *MCDU*. Finally the module *Anemometer* (*Anemo*) computes and broadcasts the speed (M) and the altitude (Z) to *AP*. Z and M are two critical data that are encapsulated in TTEthernet frames. They are transmitted in two distinct frames, which are transmitted through VL_1 from *Anemo* to *AP* via SW_1 and SW_2.

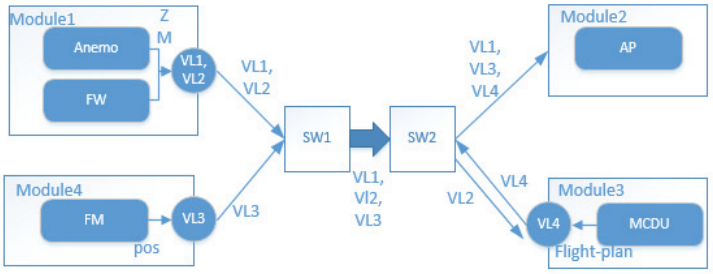

Fig. 6. The Navigation & Guidance system

3.2. Model Transformation

The model of the navigation and guidance system using our AADL extension is given in Figure 7. This is an instance of the AADL-TTEthernet metamodel discussed previously and is specified using the concrete textual syntax

that we implemented in our proposed AADL annex for TTEthernet and described in[4]. Figure 8 shows the internal representation of this model in the Eclipse EMF modeling framework and the corresponding target model (i.e. instance of the DEVS metamodel) that is generated using the model transformation step 1 shown in Figure 5. The final model used the DEVS simulation environment is shown Figure 9 and is the result of the step 2 shown in Figure 5, which consists essentially in adding the behavior of the source model to the Java code produced with Acceleo[8].

```
Annex TTEthernet {**
Module Module1
    Partition Anemo
        frames :TTE Z
    Partition FW
        frames : TTE M

Module Module2
    Partition AP
        frames : RC one

Module Module3
    Partition MCDU
        frames : RC one

Module Module4
    Partition FM
        frames : RC one

Switch  SW1
Switch  SW2

VirtualLink vl1
    Partition Anemo
        frames : TTE Z
    Partition AP
        frames : TTE Z

VirtualLink vl2
    Partition MCDU
        frames : TTE M
    Partition FM
        frames : TTE M

VirtualLink vl3
    Partition AP
        frames : RC one
    Partition FM
        frames : RC one

VirtualLink vl4
    Partition AP
        frames : RC one
    Partition MCDU
        frames : RC one   **}
```

Fig. 7. Textual Syntax

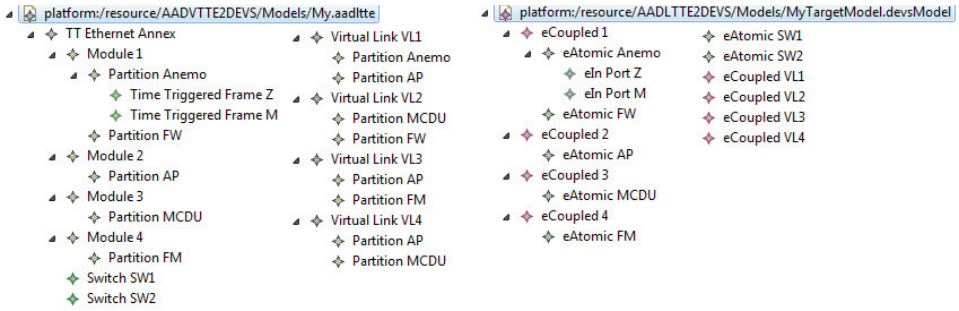

Fig. 8. TTEthernet Model (left) and the corresponding DEVS model (right)

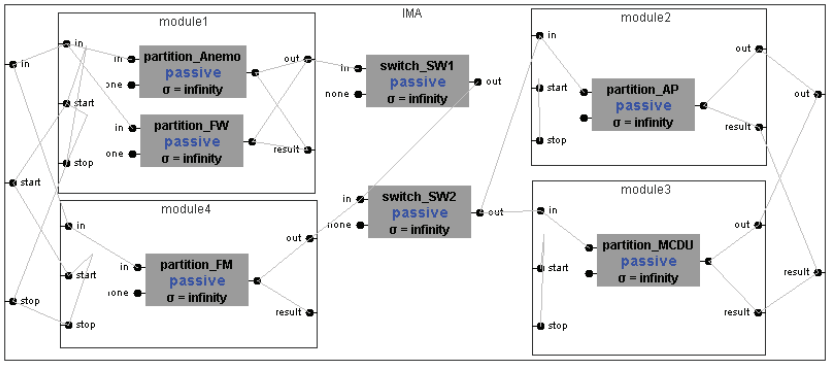

Fig. 9. Simulation graph for the navigation & guidance system

3.3. Property verification: Contention-Freedom

In this section, we illustrate the verification step in our proposed approach of the scheduling properties of TTEthernet. The schedule for a TTEthernet-based system need to meet a specific set constraints and properties defined in[10]. We consider in particular the fundamental constraint of TTEthernet network called *contention-freedom*. This constraints ensures the mutual exclusion of the frames transmitted in the same dataflow link, which means that within

a given dataflow link, only one frame can be transmitted at a certain time. Therefore, for a pair of frames transmitted within a given link, either the dispatch of the second frame should come after the end of transmission of the first one or the dispatch of the first should come after the end of the second. In order to verify the *Contention-freedom* property, we have run two scenarios where in the first scenario the schedule fulfills the contention-free constraint and the in the second the schedule violate this constraint. In the generated simulation model *Job1* and *Job2* represent respectively the TTEthernet frames Z and M in the input model of the simplified navigation and guidance system. With the first scenario, *Job1* is dispatched at instant 10 and is received by Module 2 at instant 40 and *Job2* is dispatched at instant 40 and is is received at instant 70 by Module 2. Therefore, the *contention-freedom* is verified with all jobs in the first scenario corresponding to the first scenario. However, for the schedule used in the second scenario, the dispatch time of *Job2* at instant 30 takes place before the reception time of *Job1* by Module 2 at instant 40 violating the *contention-freedom* constraints (i.e the frames Z and M transmission would be overlapping in the same dataflow link).

4. Related Work

In this section, we briefly outline the main related research work with a focus on the model transformation approach to support the verification and simulation. In Some[19], models using UML state charts are transformed into DEVS model to overcome the gap between the UML graphical modeling elements and DEVS specification. System models using the SysML modeling language are transformed into to DEVS executable models since SysML model is not simulation-specific[20]. In[18], the authors have developed simulation model using Simulation Model Definition Language (SMDL). In this work, a simulation model using DEVS is transformed to the standard SMP2.

From the perspective of AADL models verification, using the model checking techniques for this purpose tends to be very challenging[15]. AADL models are therefore often transformed into a different verification formalism. For instance, in[16], describes the translation of AADL to BIP which allows the simulation of AADL model. The transformation of AADL to timed automata is proposed in[15].

5. Conclusion and Future Work

In this paper, we have presented a model transformation based approach to enable the simulation of avionic applications deployed on distributed IMA systems with TTEthernet as a communication infrastructure. The input system models are specified using our extension to AADL modeling language to enable the modeling of IMA systems with TTEthernet. The generated models can be simulated in a DEVS simulation environment to check the model against TTEthernet constraints for instance. We have applied our approach to generate a simulation model starting with an AADL model of a simplified version of a navigation and guidance system and illustrated the verification of the compliance of the system schedule with the contention-free constraint. Currently, the automation of the refinement step of the model transformation (i.e. step 2 in Figure 5) is challenging and still requires some significant manual input from the user to fully produce the target simulation model. As a future work, we aim at addressing this limitation. In addition, we will develop further the verification of other requirements and constraints to ensure that a system model is fully compliant with TTEthernet specification.

References

1. Hessam S. Sarjoughian and Abbas Mahmoodi Markid. 2012. *EMF-DEVS modeling*. In Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium (TMS/DEVS '12). Society for Computer Simulation International, San Diego, CA, USA, , Article 19 , 8 pages.
2. Aeronautical Radio Incorporated. Avionics Application Software Standard Interface, 2013.
3. Aeronautical Radio Incorporated. ARINC Report 664P7-1 Aircraft Data Network, Part 7. AEEC, Maryland, USA, 2009.
4. Tiyam Robati, Amine El Kouhen, Abdelouahed Gherbi, Sardaouna Hamadou, John Mullins. An Extension for AADL to Model Mixed-Criticality Avionic Systems Deployed on IMA architectures with TTEthernet. ACVI at MoDELS, 2014.
5. DEVS-Suite Simulator. http://devs-suitesim.sf.net, February 2009.
6. Generation of DEVS Modeling and Simulation Environment. Ernesto Posse, Jean-Sbastien Bolduc, Hans Vangheluwe, 2003/7.
7. Bernard P. Zeigler. Multifacetted Modelling and Discrete Event Simulation. Academic Press, London, 1984.

8. Acceleo. http://www.eclipse.org/acceleo.
9. Michael Lauer, Jerme Ermont, Claire Pagetti, and Frederic Boniol. 2010. Analyzing end-to-end functional delays on an IMA platform. Volume Part I (ISoLA'10), Tiziana Margaria and Bernhard Steffen (Eds.) Berlin, Heidelberg, 243-257.
10. W.Steiner. An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks. Real-Time Systems Symposium (RTSS), IEEE 31st, 2010.
11. SAE Aerospace. 2011. Time-Triggered Ethernet, sae as6802 edition.
12. S. Beji, S. Hamadou, A. Gherbi and J. Mullins. SMT-Based Cost Optimization Approach for the Integration of Avionic Functions in IMA and TTEthernet Architectures. DS-RT '14. Washington, DC, USA.
13. Hermann Kopetz and Günther Bauer. 2003. The time-triggered architecture. Proceedings of the IEEE, 91(1):112–126, 2003.
14. Christopher B. Watkins and Randy Walter.2007. Transitioning from federated avionics architectures to integrated modular avionics. In DASC'07. IEEE/AIAA 26th.
15. M. El Hamdane, A. Chaoui and M. Strecker. From AADL to Timed Automaton - A Verification Approach. International Journal of Software Engineering and its Applications 01/2013.
16. M. Chkouri, A. Robert, M. Bozga and J. Sifaksi, Translating AADL into BIP -application to the verification or real-time systems, Models in Software Engineering, Springer-Verlag Berlin, Heidelberg, (2009).
17. Graham S. Hemingway. 2011. Time-Triggered High-Confidence Embedded Systems: Modeling, Simulation, Analysis and Back. Ph.D. Dissertation. Vanderbilt University, Nashville, TN, USA.
18. Y. Lei, W. Wang, Q. Li, and Y. Zhu. A transformation model from DEVS to SMP2 based on MDA. Simulation Modelling Practice and Theory, pages 16901709,2009.
19. Jos L. Risco-Martna Saurabh Mittalb Bernard P. Zeiglerb. Jess M. de la Cruza.From UML State Charts to DEVS State. Machines using XML. 2007 Published by Elsevier Science B. V.
20. G. Kapos, V. Dalakas, M. Nikolaidou and D. Anagnostopoulos. An integrated framework for automated simulation of SysML models using DEVS. Simulation 90, 6 (June 2014), 717-744
21. France, Robert and Rumpe, Bernhard. Model-driven development of complex software: A research roadmap. Future of Software Engineering. pp. 37–54. 2007. IEEE Computer Society
22. Mens Tom and Pieter Van Gorp. A taxonomy of model transformation. Electronic Notes in Theoretical Computer Science 152 (2006): 125-142.
23. Jouault, Frdric, Freddy Allilaire, Jean Bzivin, and Ivan Kurtev. ATL: A model transformation tool. Science of computer programming 72, no. 1 (2008): 31-39.