Time-Triggered Ethernet Metamodel: Design and Application

Tiyam Robati[1], Amine El Kouhen[1], Abdelouahed Gherbi[1]*, John Mullins[2]
[1] Department. of Software and IT Engineering, École de Technologie Supérieure, Montreal, Quebec, Canada.
[2] Department of Computer and Software Engineering, École Polytechnique de Montréal, Canada.

* Corresponding author: Email: abdelouahed.gherbi@etsmtl.ca
doi: 10.17706/jsw.11.10.1040-1053

Abstract: The combination of the SAE Time Triggered Ethernet (TTEthernet) standard with the Integrated Modular Avionics (IMA) architectures supports the design, deployment and integration of mixed-critical avionic applications. In order to cope with the complexity of these tasks, we advocate for a model-driven engineering methodology. The key element of such methodology is the modeling language, which enables producing relevant models of the system. In this paper, we present a metamodel, which captures the main features and concepts defined in the SAE TTEthernet standard. We discuss how a combination of the TTEthernet metamodel with an IMA metamodel can be used to extend the AADL modeling language to model avionic applications deployed a TTEthernet-networked IMA platform. Finally, we present a case study to illustrate our approach.

Key words: MDE, ttethernet, IMA, AFDX, metamodel.

1. Introduction

Safety-critical systems should meet strict safety, reliability and performance requirements because of their important impact on human life, economy and environment. An example of this class of systems is the avionic systems, which are increasingly developed independently and integrated following the principles of the Avionic Modular Avionic (IMA) architecture.

An IMA-compliant infrastructure is a distributed system interconnected using a suitable networking technology. The Ethernet networking standard has been extended to support the performance requirements of avionic systems. These extensions are Avionic Full Duplex AFDX standard ARINC 664 [1] and more recently the SAE AS6802 (TTEthernet) [2]. The latter in particular is a *Quality-of-Service (QoS)* enhancement for Ethernet networks in order to achieve deterministic, synchronous, and congestion-free communication paradigm [2].

The combination of IMA and TTEthernet provides an infrastructure, which supports the integration of distributed avionic applications having different criticality levels leading to mixed-criticality systems. The integration of avionic functions developed independently is however a complex engineering task. In order to cope with such complexity, the principles of the Model-Driven Engineering (MDE) approach are effective. This is because the MDE approach is essentially based on building models of the system and their manipulation through model transformations to enable applying diverse relevant analysis techniques and eventually to help in generating implementations (i.e. code). The key element of the MDE approach is the modeling language that can be used in order to express the system at a convenient level of abstraction and to interface it with suitable formal analysis techniques to verify safety and performance properties of the system.

In this paper, we present and discuss a metamodel that the captures the concepts and main features of TTEthernet as described in the SAE AS6802 standard [2]. We show how we leverage this TTEthernet metamodel to define an extension of the Architecture Analysis & Design Language (AADL) to support the modeling of the concepts and constraints relevant for TTEthernet standard. We discuss the implementation of this extension and illustrate it with a concrete example.

This paper is organized as follows: In Section 2, we present an overview of the main features and concepts of TTEthernet standard. We present and discuss in Section 3 our proposed metamodel to capture the concepts of TTEthernet. We discuss in Section 4 how the proposed metamodel can be used to support the modeling of avionic applications. In Section 5, we show how we leverage the TTEthernet metamodel to extend the AADL modeling language to model avionic applications deployed on TTEthernet-networked IMA platform. We illustrate our approach an illustrative example in Section 6. In Section 7, we succinctly review the most close related research works to ours. Finally, we conclude the paper and outline our ongoing and future research work in Section 8.

2. Overview of the Time-Triggered Ethernet Standard

TTEthernet provides time-triggered services on top of the basic Ethernet features in order to allow synchronous communication with constant latency, tight jitter (μsec) and determinism requirements. TTEthernet integrates three types of data flow: Time-Triggered (TT) data flow which is the highest priority traffic; Rate Constrained (RC) traffic, which is equivalent to AFDX traffic, and finally Best Effort (BE) traffic. As a consequence, TTEthernet supports mixed-criticality applications where highly critical functions cohabit with less critical functions.

TT frame represents a frame that should be transmitted at specific time instants. A synchronized global time is established allowing to compute an off-line schedule, which specifies the dispatch frame point in time and temporal characteristics for intervals used for asynchronous traffic such as RC and BE.

The temporal properties of TT frame f_i are specified by equation (1):

$$f_i[v_x, v_y] = f_i.period,\ f_i^{[v_x, x_y]}.offset,\ f_i.length \tag{1}$$

The period and length of frame are the off-line configured parameters of system, and the offset is assigned by scheduler. The offset for a frame F on a link L in the network is: F^L.*offset*.

The dispatch point in time of a TT frame f_i, on communication link $[v_x, v_y]$, is denoted by: $f_i^{[v_x, v_y]}$ is identified by the period and offset of frame where $f_i^{[v_x, v_y]}$ represents frame f_i transferred as TT on a communication link $[v_x, v_y]$.

The RC traffic, which corresponds to the AFDX traffic [1], guarantees bounded latency in complex network. This is achieved through a sharing of the network bandwidth between functionalities of system and maintaining the predictability of the communication [3]. The characteristics of an RC frame are max transmission rate and its length, described by equation (2):

$$f_i = f_i.rate,\ f_i.lenght \tag{2}$$

The RC frame should always respect its transmission rate limit. In the case where an RC frame exceeds its transmission rate, a traffic policing function implemented in the TTEthernet switch (e.g. leaky bucket) drops this frame. The traffic policing function measures the time between two frame receptions to monitor the transmission rate.

Finally, the best effort (BE) traffic represents the classic Ethernet approach, where no guarantee exists for the transmission time, reception at the recipient location and delays. In fact BE frame uses the remaining bandwidth of the network.

TT traffics must be free of any conflict. To do so, schedule of the frames transmission with respect to the synchronized global time should be established off-line. This is a fundamental constraint of TTEthernet network called *contention-freedom*. It 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. Off-line schedule planned at the system design time, is responsible for prohibiting runtime conflict. Therefore in the schedule, TT frame has higher priority than RC and BE. Once a TT frame and a RC frame arrive in same outgoing port, the TT frame takes priority over the RC frame. In fact RC traffics are dispatched if TT traffic is not pending. Therefore when TT traffic arrives, should be immediately transmitted. To insure the immediate transmission, switch must confirm freedom of network.

TTEthernet is a transparent synchronization protocol because of coexisting of different traffic types on the same physical communication network. In fact, this synchronization protocol permits transparent integration of time-triggered services on top of standard Ethernet infrastructure.

TTEthernet introduces a fault-tolerant algorithm which detects failures and disorders in the network. In particular, these fault-tolerant algorithms set up the send order of synchronization message (i.e. *PCF*s) in order to ensure synchronization of local clocks in a distributed system. *Protocol Control Frame (PCF)* is a dedicated Ethernet frame which carries TTEthernet protocol control frame for the reason of local clock synchronization. Toward that reason, a multitude of TTEthernet End-Systems generate *PCF*s and distribute them with TTEthernet switches.

Fault-tolerant algorithms use multiple redundant paths established by TTEthernet network, in order to tolerate failure of a single path without affecting the entire application of system. This is vital for safety-critical system that is achieved by the fault-tolerant algorithms, where multiple redundancies causes that multiple faults are tolerated.

As previously mentioned, TTEthernet provides local clock synchronization in distributed systems. To do so, a synchronization approach is planned. The main elements of synchronization approach are*: Synchronization Master (SM)*, Compression *Master (CM)* or *Synchronization Client (SC)*.

The components of system can be selected as *SM* and *CM* based on the requirements on the system architecture. Once the system designer decided about the configuration of *SM* and *CM*, the remaining components are configured as *SC*.

The synchronization approach of TTEthernet is organized in two steps. In the first step *SM*s send *PCF*s to the *CM*s. Then after new calculation a new *PCF* is sent out from *CM*s to *SM*s and *SC*s as second steps of approach. The new *PCF* contains an averaging value of arrival times of dispatched PCFs in first step.

Synchronization topology is established in different level of the system architecture. The lowest level of this topology composed of End-Systems and switches which are configured as *SM*, *CM* and *SC*. The next level presents the concept of *cluster* where single *synchronization domain* and *synchronization priority* is considered.

TTEthernet introduces different *synchronization domains* and *synchronization priorities* in order to native support of system-of-system communication. *Synchronization domains* specify independent TTEthernet systems inside of a system-of system with respect to their *synchronization priorities*. It is important to mention that two components belong to different synchronization domains will never synchronize their local clocks. That means the Communication between two components of different synchronization domains is only possible with non-time-triggered traffic classes.

The concept of *cluster* is defined in TTEthernet to permit running in isolation different *cluster*s in a large TTEthernet network. A cluster is organized as a set of End-Systems and switches that are connected together using communication redundancy channel. The communication channel at least contains one switch.

Several clusters build a *multi cluster* in next level of synchronization topology where one *synchronization domain* and many *synchronization priorities* are introduced. A *multi cluster* system supports a master-slave paradigm, which try to synchronize all devices in system toward the highest synchronization priority.

Finally, *network* level composed of several *multi cluster*, different *synchronization domains* and *synchronization priorities.*

This is important to mention that dataflow between two TTEthernet *cluster*s in different *synchronization domains* is supported by RC or BE traffic.

3. Metamodel for TTEthernet

In this section, we describe the main concepts of the metamodel we propose to support modeling distributed systems using TTEthernet as a communication infrastructure. These concepts correspond to the main features of TTEthernet defined in the SAE TTEthernet standard, AS6802 [2]. This metamodel is designed to support building a set of tools to perform the design and analysis of TTEthernet-based distributed architectures. The overall view of our TTEthernet metamodel is depicted in Fig.1.

Fig. 2 introduces the overview of TTEthernet metamodel, based on this figure, *TTEthernet Metamodel* class is composed of *Processing Resource*, where each *Processing Resource* could be *Synchronization Master*, *Compression Master* or *Synchronization Client*. In other hand, *Processing Resources* are divided into two classes, *Networking Resource* such as *Switch* and *Computing Resource* such as *End-System.* In high level of abstraction, the presented classes of TTEthernet metamodel satisfy the requirements for modeling TTEthernet *cluster.* Remember that a *cluster* is composed of at least two *computing resources* that communicate together through a *networking resource*. In fact, the only missing part for modeling a *cluster* is, how the *networking Resources* and *computing Resource*s are connected together. This is covered in Fig. 3, where we present the concept of *virtual link*. Briefly, virtual link is a logical link connection from one source End System to one or more destination End Systems. We explain it more in details later on this paper.

The *Synchronization Domain* class of TTEthernet metamodel provides a specific domain of synchronization for a cluster. Every *synchronization domains* have its unique priority presented by the *Synchronization Priority* class of the metamodel. *Cluster* supports only one *synchronization Priority* and one *Synchronization Domain*. A *multi cluster* that is composed of at least two *cluster*s, presents one *Synchronization Domain* and multiple *Synchronization Priorities*. It is supported by *EReference* between cluster *class* and *Synchronization Domain* of the metamodel. Finally, the TTEthernet *network* is supported by *TTEthernet MetaModel* class, where it can have multiple *Synchronization Domains* and multiple *Synchronization Priorities.*

After presenting the main elements of TTEthernet metamodel, in following we take a deeper look at other essential elements of the metamodel. Fig. 3 illustrates the *Schedulable Resources* of TTEthernet network that represents all the elements that related to scheduler. These resources are: *End System*, *Frame*, *channel* and *virtual link*. *End-System*s are the nodes of distributed system that perform the functionality of system. To do so, every *End-System* hosts at least one *Functionality* of system. This is supported by *EReference* between *End-System* class and *Functionality* class of the metamodel. In case that *End-System* hosts multiple *Functionalities*, the strict isolation between them is required. The maximum number of *Functionality* that could be dedicated to an *End-System*, is determined by resource allocation function.

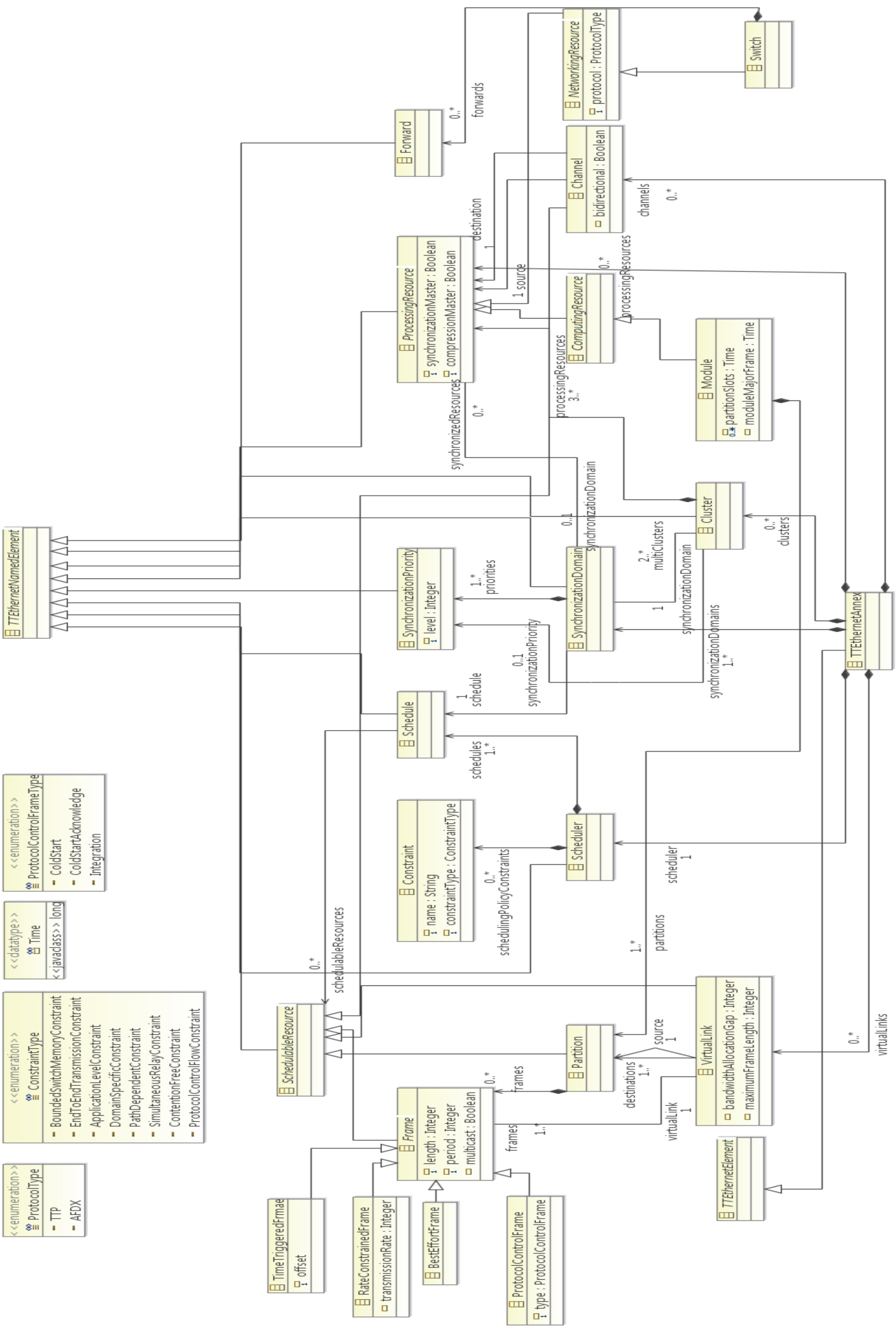

Fig. 1. TTEthernet metamodel.

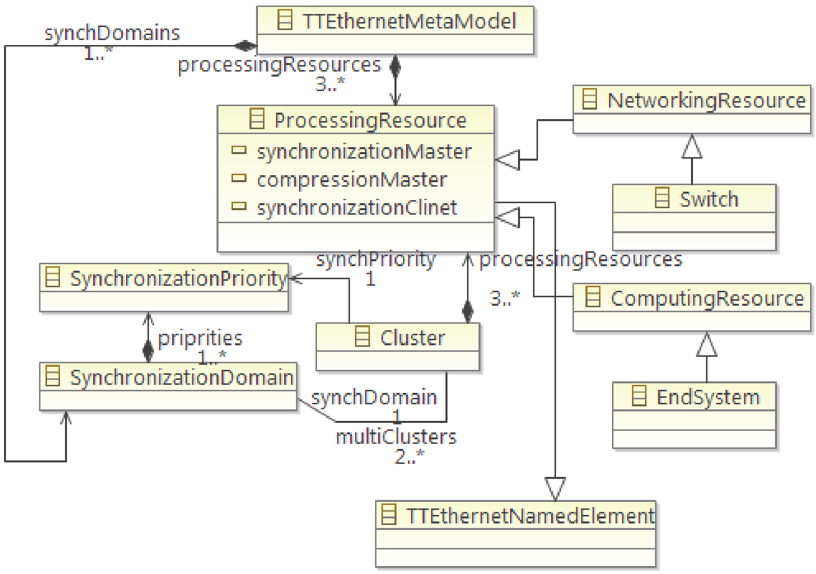

Fig. 2. Main components of the TTEthernet metamodel.

Frames are the data unit that traveled through the network. *Virtual Link* is a logical connection that builds communication tree between one *End-System* as source and multiple *End-Systems* as destination.

Each *Frame* belongs to one *Virtual Link*, where *Virtual Link* carries many *Frames*. This is shown by *EReference* between *Frame* class and *Virtual Link* class of metamodel. Each *Functionality* must have only one *Virtual Link* as its source and one or many as its destination. This is supported by bi-directional *EReference* between *Functionality* class and *Virtual Link* class.

Channel is a logical connection defined within the scope of a *cluster* or *multi cluster.* In fact *Channel* is provided to elaborate the communication between clusters and multi clusters. *Frames* belong exclusively to one channel, but the *End-Systems* and its dedicated *Functionalities* could use one or more channel.

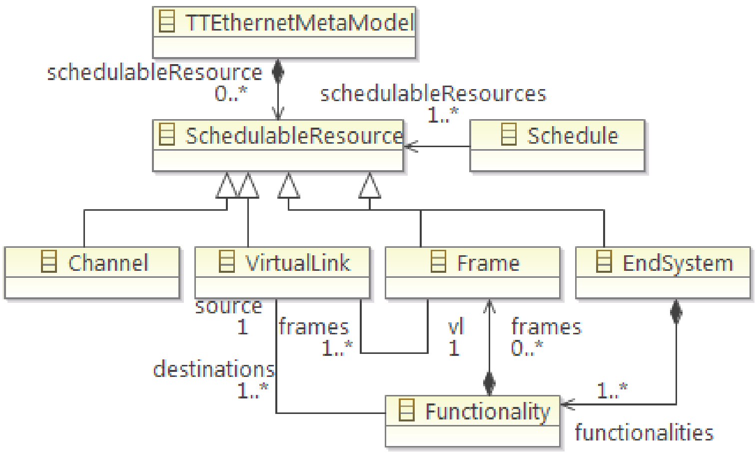

Fig. 3. Schedulable resources.

As it is mentioned in background section, TTEthernet presents three types of frame with different priority depicted in Figure 4. The attributes of each class of the metamodel in this figure, present the characteristics of corresponding to frame type (e.g. *TTEthernet* class has *offset*). *PCF* class which is a frame type too, supports requirements of Synchronization protocol of TTEthernet.

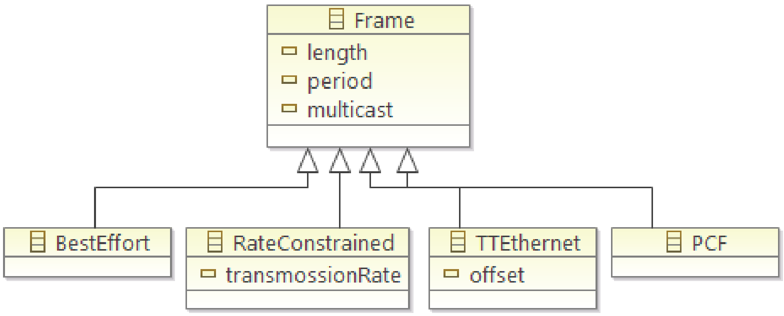

Fig. 4. Frame categories.

A description of the temporal communication behavior of frame is provided by *Schedule*. *Scheduler* is a tool that produces *Schedule*, which should respect specific constraints in order to support TTEthernet communication paradigm. Figure 5 illustrates *Schedule* and *Scheduler* classes of metamodel including enumeration list of presented constraints. These constraints that have been reported in [4] are: *Contention Free, Simultaneous Relay, Path Dependent, Domain Specific, Application Level, End To End Transmission, Bounded Switch Memory and Protocol Control Flow*. TTEthernet provides unique *Schedule* overall network even if the network is constructed of several *multi clusters* and *Synchronization* Domains.

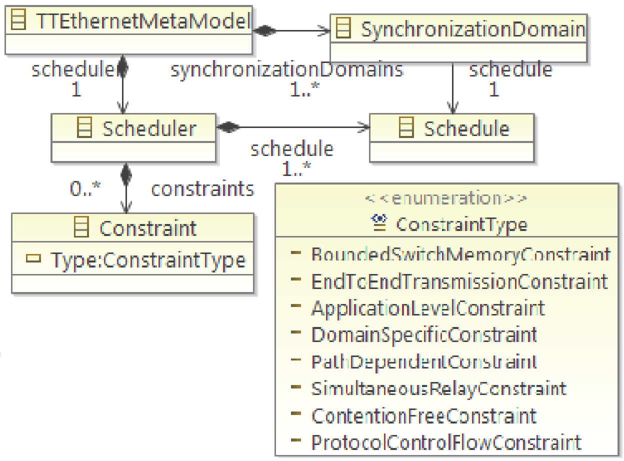

Fig. 5. TTEthernet scheduler.

4. Using the TTEthernet Metamodel to Support IMA-Based Avionic Systems

The Integrated Modular Avionics (IMA) is a distributed architecture which is designed to support resource sharing between functionalities while ensuring their isolation in order to prevent interference between them [5]. Consequently, this reduces the cost of large volume of wiring and equipment and keeps up with strict safety requirements. IMA architecture supports the space and time partitioning using of ARINC 653-compliant real-time operating system [6]. The partitioned environment of IMA allows hosting different avionics functions with different criticality levels (e.g. control functions and comfort functions) on the same platform.

The IMA architecture is composed of End-Systems (ES) called modules and switches (SW) connected together through the communication links. The communication network of IMA architecture can be either ARINC 664 [1] or the TTEthernet [2]. TTEthernet has a higher determinism level and suites better the requirements of safety-critical systems. This is because TTEthernet supports cohabiting high critical functions with less critical ones on the same physical platform. In this section, we discuss how the TTEthernet metamodel described in the previous section support the modeling TTEthernet-networked

IMA-bases avionic systems. Therefore, we discuss in the following a particular profile of the TTEthernet metamodel which integrates the concepts relevant for IMA architecture.

The two main components of IMA architecture are End-System (ES) also called modules in ARINC 653 [7] and the switches (SW). The modules are functional execution units of IMA architecture. The communication between the modules is performed by switches and communication links. The partitioned environment of IMA is supported by the partitions deployed on modules. A partition has a strict access to processing resources and memory. The physical links of the network connect the modules through a set of switches and support the concept of data flow link. A sequence of directed data flow link performs a data flow path. Fig. 6 depicts the metamodel of IMA architecture.

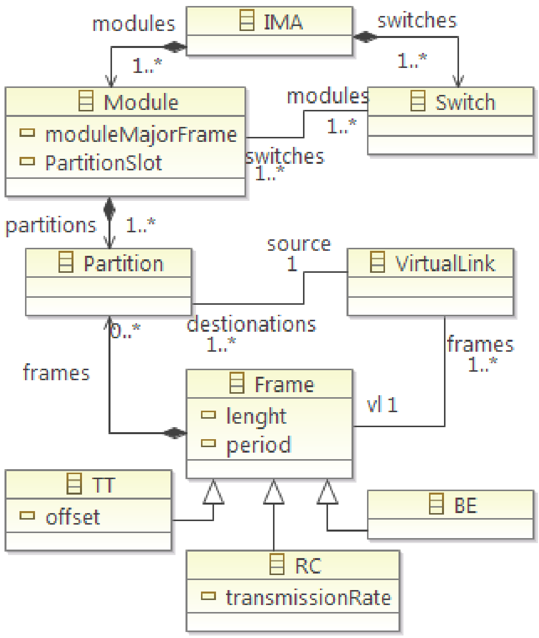

Fig. 6. The Metamodel of IMA.

Table 1 demonstrates how IMA metamodel presented in Figure 6 can be mapped to TTEthernet metamodel concepts. The TTEthernet metamodel provides the concepts which correspond to the main components of IMA metamodel and then provides a model that represents IMA architecture using TTEthernet.

Table 1. IMA Metamodel in Accordance with TTEthernet Metamodel

IMA:	TTEthernet:
IMA	Cluster
Module	End-System
Partition	Functionality
Frame	Frame
Switch	Switch
VL	VL

5. TTEthernet Metamodel Application: Extending AADL

AADL is a standard architecture description language developed by SAE AS5506 [1], [8] for the formal specification of the hardware and software architecture of embedded computer systems. It is used for the modeling of software system architectures and supports the analysis and verification non-functional properties of modeled system. The AADL modeling language can be extended through two built-in extensibility mechanisms. The first mechanism is a construct for property set definition. This construct enables defining or modifying AADL properties. The second extensibility means is an annex extension mechanism, which enables to specify sub-languages that will be processed within an AADL model. Some AADL annexes are now standardized including for example is the Error Modeling Annex [9], which allows specification of error models to be associated with core components supporting safety and dependability modeling. We use the second approach to extend the AADL modeling language with the features defined in the TTEthernet metamodel. In order to do so, we first define a concrete textual syntax for the TTEthernet metamodel which will be used by the system engineer to describe its model. The sublanguage defined by the TTEthernet metamodel is then included into the AADL as annex subclause. The complier of this TTEthernet sublanguage concrete syntax is then integrated with the OSATE2 tool environment using its extension points. This integration is shown in Fig.7. More details on this implementation can be found in [10].

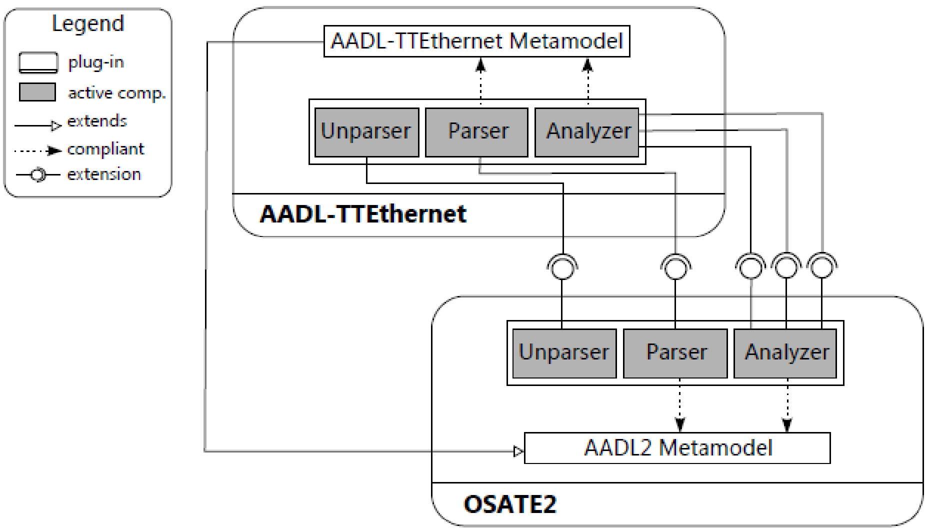

Fig. 7. TTEthernet sublanguage integration within OSATE2 tool.

6. Case Study

Our purpose in this section is to illustrate the proposed approach with a representative example of an avionic system. We have modeled this system using the AADL extension described in the previous section. The system shown in Figure 8 is composed of eight modules and four switches (SWs). The modules host different number of partitions as follows:

- The module *ES1* hosts the partitions *Nose Landing Gear (NLG)*, *Gauging Channel A (GCA)* and *Fuel Mass (FM).* Fuel Gauging provides the measurement method for the physical characteristics of the fuel such as temperature, density and permittivity. Using these data the system computes the volume and the mass of fuel available in each fuel tank.
- The module *ES2* hosts the partitions *Main Landing Gear (MLG), Gauging Channel B (GCB)* and *Fuel Transfer Center (FTC). Nose Landing Gear (NLG)* provides commands for the *NLG* extension/retraction to actuators. *Right and left Main Landing Gear (MLG)* provides commands for the *LMLG* extension/retraction and for the *RMLG* extension/retraction to actuators.
- The module *ES3* contains the partitions *Flight Deck Emulator (FDE)*, *LG & Fuel Aircraft Emulation (LGFAE)* and *SP Aircraft Emulator (SPAE)*, where *LGFAE* emulates the behavior of actuators and

provides status to the *NLG* and *MLG*.

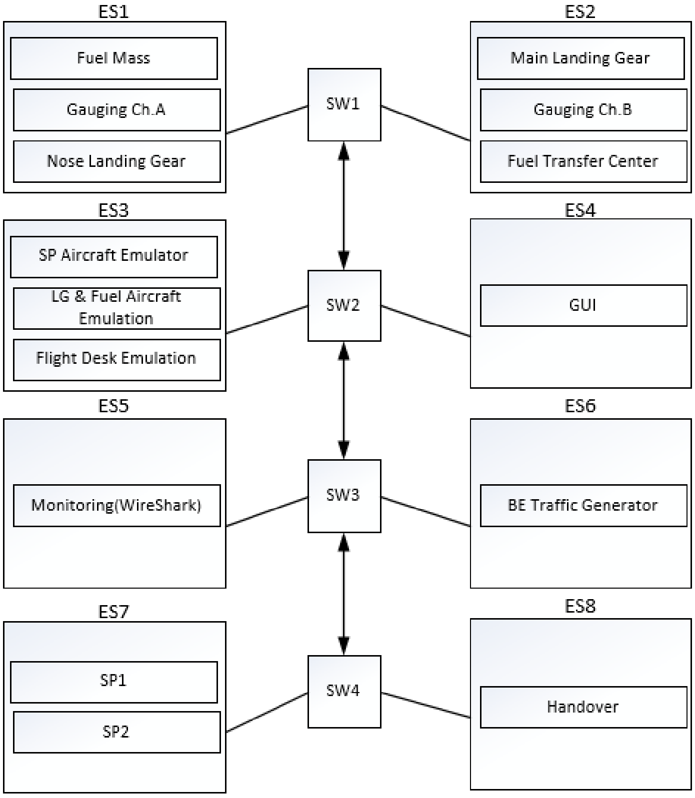

Fig. 8. Case study.

The communication between modules (i.e. ESs) is performed through the large number of *Virtual Links (VL)*. Table 2 gives the properties of VLs, which are mainly the source and destination partition, BAG and the max size of frames. For the sake of simplicity, we present some of these *VLs* partially. The model of the presented case study using our AADL extension is given in Fig. 9.

Table 2. Virtual Links Details

VLID	Source	Destination	BAG	Max Frame Size
0	GUI	FDE	15	114
13	GCB	FM	15	562
14	FM	FTC	15	562
15	FTC	LGFAE	15	114
42	NGL	LGFAE	15	114
43	MLG	LGFAE	15	114
140	SPAE	SP1	1	1518
1000	GCA	FDE	1	1518

7. Related Work

We present in this section a review of the main related research work to our work presented in this paper. An extension to the OMNeT++ INET framework is developed to support simulation of TTEthernet by [11]. Authors of [12] introduce and develop a TTEthernet model using SystemC/TLM in order to facilitate the design and integration of Cyber Physical System (CPS). Moreover, [13] introduces a TTEthernet simulation environment based on OPNET for generic building blocks such as switches and systems. The research reported in [14], [15] and [16] propose a modeling approach, which describes different levels of detail of IMA execution platform interconnected with AFDX. . In [15] authors define a modeling approach based on AADL and SystemC, which aims at the design and dynamic simulation of an IMA-based avionics platform. This is component-based approach, which can be used to dimension the architecture taking into consideration the application to be deployed while achieving early platform validation.

Lauer et al. [17] propose a modeling approach that computes worst case traversal time (WCTT) for an IMA architecture interconnected with AFDX. They produce functional analysis using model-checking verification approach. The authors develop in [18] worst case temporal consistency, latency and freshness analysis for IMA platform with different evaluation method such as tagged signal model and Integer Linear Programming (ILP). Lauer *et al.* [19] focus on IMA architecture using TTEthernet to present a cost optimal strategy for the integration of multi-critical function in IMA architecture. This work also used binary integer problem formalization using off-the-self solver.

```
Annex TTEthernet {**

Module ES1
    Partition FM
        frames : TTE TTE1
                Offset : 15
                Length : 1
                Period : 1

    Partition GCA
        frames : TTE TTE2
                 Offset : 15
                 Length : 1
                 Period : 1

    Partition NLG
        frames: TTE TTE3
                Offset : 15
                Length : 1
                Period : 1
Module ES2…
Module ES3…

Module ES4…

Module ES5…

Module ES6…

Module ES7…


Module ES8…
```

```
Switch  SW1
Switch  SW2
Switch  SW3
Switch  SW4

VirtualLink vl0
    Partition GUI
        frames : TTE TTE8
                 Offset : 15
                 Length : 1
                 Period : 1
    Partition FDE
        frames : RC RC2
                 BAG : 15
                 Length:1
                 Period:1

VirtualLink vl13
    Partition GCB…
     Partition FM…

VirtualLink vl14
    Partition FM…
    Partition FTC…

VirtualLink vl15
    Partition FTC…
     Partition LGFAE…

VirtualLink vl42
    Partition NGL…
     Partition LGFAE…
```

```
VirtualLink vl43
    Partition MGL…
     Partition LGFAE…

VirtualLink vl140
    Partition SPAE…
     Partition SP1…

VirtualLink vl1000
    Partition GCA…
     Partition FDE…

              **}
```

Fig. 8. The system AADL partial model using TTEthernet annex .

Several AADL extensions based its built-in extension mechanisms are now standardized as official annexes. These include the Data modeling annex, ARINC653 annex, the AADL Behavior Annex [7], and Error Model Annex [9]. In addition, some research work have focused on extending the language using these extension mechanisms or investigating alternative ways. The most close research works to ours from this perspective are reported in [20] and [21]. Delange *et al.* [20], present an approach based on AADL, which covers the modeling, verification and implementation of ARINC653 systems. The authors describe in the work the modeling guidelines elaborated in the ARINC653 annex of the AADL standard. This approach is supported by a tool chain composed of Ocarina AADL tool suite, AADL/ARINC653 runtime POK and Cheddar scheduling tool. The authors of [21] present an implementation of the AADL behavior annex as an extension plug-in to the OSATE 2. We have implemented our AADL TTEthernet extension using similar techniques. De Niz and Fieler discuss in [22] how to extend the AADL language to include new features for the separation of concerns (i.e. Aspects). Based on this research work, it seems that the AADL extension mechanisms do not support the separation of concerns and new aspect-like constructs and mechanisms are then investigated. Brau et al. [3] present a model of a subsystem of Flight Management System using AADL and show how to establish important parameters in the AADL model including the virtual links characteristics for instance.

8. Conclusion

The aim of this work is to support the modeling of mixed-criticality avionic application, which are deployed a platform composed of a distributed IMA architecture with TTEthernet as a networking infrastructure. This modeling support would help in coping with the inherent complexity of such engineering task. This can be achieved with a model-driven engineering methodology. As a building bloc of this methodology, we presented in this paper the design of the TTEthernet metamodel. We have used this metamodel with a mapping to the IMA components to extend the AADL modeling language to support the aforementioned goal.

Acknowledgment

This work is partially supported by the CRIAQ-NSERC RDC Project VerITTAS (Verification and Integration of multi-critical Time-Triggered Avionics Systems) (AVIO613) No. 435325-12.

References

[1] *Aeronautical Radio Incorporated. ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*. AEEC, Maryland, USA.

[2] SAE Aerospace. *Time-Triggered Ethernet*.

[3] Brau, G., Hugues, J., & Navet, N. (2013). Refinement of aadl models using early-stage analysis methods: An avionics example. *Technical Report TR-LASSY-13-06, Laboratory for Advanced Software Systems*.

[4] Steiner, W. (2010). An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks. *Proceedings of the 31st IEEE Real-Time Systems Symposium*.

[5] Watkins, C. B., & Walter, R. (2007). Transitioning from federated avionics architectures to integrated modular avionics. *Proceedings of the 2007 IEEE/AIAA 26th Digital Avionics Systems Conference Digital Avionics Systems*.

[6] *Aeronautical Radio Incorporated. ARINC Report 653P0 Avionics Application Software Standard Interface*, Part 0, Overview of ARINC 653.

[7] SAE Aerospace. SAE Architecture Analysis and Design Language (AADL) Annex Volume 2: Annex B: Data Modeling Annex Annex D: Behavior Model Annex Annex F: ARINC653 Annex.

[8] *The SAE Architecture Analysis Design Language*, AS5506B. SAE Aerospace.

[9] *SAE Aerospace*. SAE Architecture Analysis and Design Language (AADL) Annex Volume 1: Annex A: Graphical AADL Notation, Annex C: AADL Meta-Model and Interchange Formats, Annex D: Language Compliance and Application Program Interface Annex E: Error Model Annex.

[10] Tiyam, R., Amine, E. K., Abdelouahed, G., Sardaouna, H., & John, M. (2014). An extension for AADL to model mixed-criticality avionic systems deployed on IMA architectures with TTEthernet.

[11] Steinbach, T., Kenfack, H. D., Korf, F., & Schmidt, T. C. (2011). An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*.

[12] Zhang, Z., & Koutsoukos, X. (2013). Modeling time-triggered ethernet in systemc/TLM for virtual prototyping of cyber-physical systems. embedded systems: Design, analysis and verification. *Proceedings of the 4th IFIP TC 10 International Embedded Systems Symposium*.

[13] Abuteir, M., & Obermaisser, R. (2013). Simulation environment for time-triggered ethernet. *Proceedings of the 11th IEEE International Conference on Industrial Informatics*.

[14] Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2000). Science of computer programming. *Annual Report 1999-2000*.

[15] Michaël, L., David, F., Marc, G., & Laurent, P. (2010). A new modeling approach for IMA platform early validation. *Proceedings of the 7th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*.

[16] Lafaye, M., Gatti, M., Faura, D., & Pautet, L. (2011). Model driven early exploration of IMA execution platform. *Proceedings of the 2011 IEEE Digital Avionics Systems Conference (DASC), In Digital Avionics Systems Conference*.

[17] Lauer, M., Ermont, J., Boniol, F., & Pagetti, C. (2010). Analyzing end-to-end functional delays on an IMA platform. *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I (ISoLA'10)*.

[18] Lauer, M., Ermont, J., Boniol, F., & Pagetti, C. (2011). Latency and freshness analysis on IMA systems. *Proceedings of the 2011 IEEE 16th Conference on Emerging Technologies and Factory Automation*.

[19] Lauer, M., Mullins, J., & Yeddes, M. (2013). Cost optimization strategy for iterative integration of multi-critical functions in IMA and TTEthernet architecture. *Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*.

[20] Delange, J., Pautet, L., Plantec, A., & Kerboeuf, M., Singhoff, F., & Kordon, V. F. (2009). Simulate, and implement arinc653 systems using the aadl.

[21] Lasnier, G., Pautet, L., Hugues, J., & Wrage, L. (2011). An implementation of the behavior annex in the aadl-toolset osate2.

[22] Niz, D., & Feiler, P. H. (2007). Aspects in the industry standard aadl. *Proceedings of the 10th international Workshop on Aspect-Oriented Modeling*.

Tiyam Robati is completing her PhD at Software and IT Engineering Department of the the Ecole de Technology in Montreal. Her research focus is mainly on the modeling of avionics software systems such as IMA systems. She graduated from Concordia University of Canada, Montreal with Master degree in in Electrical and computer engineering in 2010 and her bachelor degree of Electrical engineering in Iran in 2004. Tiyam is now a Software specialist at CAE inc.

Amine EL Kouhen was a research associate in the NSERC/Ericsson Industrial Research chair in Model-Driven Software Management. He is also a former researcher at CEA LIST (an arm of the French Atomic Energy Commission, http://www-list.cea.fr/gb/index_gb.htm) within the LISE (Laboratory for Model-based Engineering of real-time and embedded (RT/E) systems). He participated in the development of Eclipse Papyrus UML tool by improving its diagram editors. He is a Ph.D in Computer Sciences from University of Lille (North of France) and M.Sc. in Information Systems from University of Lorraine (Nancy - France). He works now as a consultant in several companies in europe (gouvernement, pharmaceutic industry...) in several areas such as Model-Driven Engineering

(MDE), programming language design, domain-specific languages, language implementation, software engineering tools.

Abdelouahed Gherbi is an associate professor at the École de Technologie Supérieur (ETS) in Montreal, Canada. He received his Ph.D. degree in computer engineering from Concordia University, Canada. He has spent one year as visiting researcher at the Defense Research and Development Canada in Valcartier, Quebec. His research interests are mainly in model-driven software engineering, modeling and analysis techniques for real-time and critical software systems, software performance, high availability and security.

John Mullins is a professor in the Department of Computer and Software Engineering at the École Polytechnique de Montréal since 1999 where he currently heads the Design and Realization of Complex Systems Laboratory (CRAC). He is also associate researcher in the Combinatorics and Mathematical Computer Sciences Laboratory (LaCIM), a research center associated with the Mathematics Department, Computer Sciences Department and the Canada Research Chair in Algebra, Combinatorics and Mathematical Computer Sciences of the Université du Québec à Montréal. After achieving a B.Sc., an M.Sc. and a Ph.D. scholarship in pure mathematics and an M.Sc. in theoretical computer science at the Université de Montréal, John Mullins has obtained a Ph.D. on recursion-theoretic models for Concurrency in telecommunication software engineering from the Institut National de la Recherche Scientifique (INRS) in 1992.