From Safety Models to Security Models: Preliminary Lessons Learnt

Pierre Bieber, Julien Brunel

ONERA-DTIM
2 Av Edouard Belin, BP 74025, F-31055 Toulouse
Firstname.Lastname@onera.fr

Abstract. We aim at developing common models and tools to assess both safety and security of avionics platforms so we studied the adaptation of models devised for Safety assessment in order to analyse security. In this paper, we describe a security modelling ana analysis approach based on the AltaRica language and associated tools, we illustrate the approach with an avionics case-study. We report lessons learnt about the convergence and divergence points between security and safety with respect to modelling and analysis techniques.

1 Introduction

Taking into account information security risks is a relatively new task in the development of safety-critical aircraft systems. Recent transport aircraft including Airbus A380 and Boeing B787 contain a security architecture that organizes the avionics platform in domains: aircraft control, airline information services, passenger information and entertainment Services. Security mechanisms such as firewalls and digital signature infrastructure are in place in order to control information flows and applications that belong to these domains.

In parallel to the development of these security architectures, an international effort has led to the creation of Airworthiness Security Process (AWSP) document ED-202/DO-326 [10] that standardizes the development process of aircraft systems with respect to security. This document aims at providing a joint basis for the certification of information security aspects of aircraft systems. Consequently, this document focuses on security aspects that have an effect on the safety of the aircraft, these aspects are called "Security for Safety". The document does not deal with other security aspects concerning, for instance, the protection of passenger privacy or the protection of aircraft manufacturer intellectual property. In this paper we restrict ourselves to the modelling and analysis of "Security for Safety".

The first generation of security architecture has to evolve in order to deal with new services for airlines such as remote maintenance or paper-less cockpit. An important goal for the design of new security architectures is to keep the costs of their implementation and assessment of security architecture as low as acceptable.

In the past decade an approach has been defined to help assess efficiently the safety of systems. This approach, called Model Based Safety Assessment [4], is based on the use of formal models of aircraft systems and of associated tools to automatically perform parts of the safety assessment required in the airworthiness certification process.

In this paper we describe an attempt to adapt the Model Based Safety Assessment approach in order to deal with Information Security aspects. We believe that the reuse of safety models and assessment tools should reduce the cost of security assessment. In the following of the paper, we first summarize the main aspects of the Model Based Safety Assessment methodology. Then we explain the adaptation of models to deal with security and we describe how we used two safety assessment tools to perform security analysis. Finally we list several preliminary lessons learnt.

2 A Summary of Model Based Safety Assessment

2.1 Safety Model

Model Based Safety Assessment aims at supporting the Preliminary System Safety Assessment (PSSA) [8]. Before the PSSA is performed, the Functional Hazard Analysis identifies the Failure Conditions (e.g. safety critical situations of the system) and assesses their severity on a scale going from No Safety Effect (NSE) to Catastrophic (CAT). Then, during the Preliminary System Safety Assessment, safety models (or alternatively fault-trees) are built and analysed. A safety model describes formally in which node a fault occurs and how this fault propagates inside the system architecture in order to cause a Failure Condition.

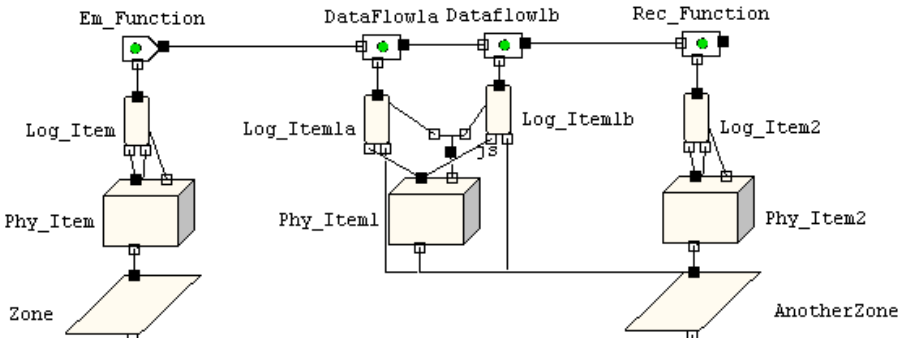

Fig. 1. Layered Architecture Model

As shown in the previous figure the safety model is organised into several layers :

– Functional Layer : at the top of the layers are depicted functional nodes, Em_Function is a function that emits a data flow (represented by the nodes DataFlow1a and DataFlow1b) towards a receiver function called Rec_Function. The links relating functional nodes represent the routing of the data-flow from an emitter to the receiver. All these nodes use resources from the logical layer. The links between the logical and functional layers connect functional nodes with the logical resources that they use.
– Logical Layer: this layer groups logical nodes such as software, partitions, virtual links or network messages that implement the functional nodes. All these nodes use resources from the physical layer. The links between the logical and physical layers connect logical nodes with the physical resources that they use.
– Physical Layer: this layer groups physical nodes such as computers, mass-memory storage, network communication equipment and links. They are used to implement the logical nodes. All these nodes are located in a Zone. The links between the zone and physical layers connect physical nodes with their installation zone.
– Zone Layer: this layer describes the various installation zone of interest: cockpit, avionics bay; cabin, aircraft vicinity, airport, maintenance operation center, ...

To implement the Model Based Safety Assessment approach we have been using the AltaRica language and associated tools [2]. Each of the nodes in the previous diagram is modelled formally by an AltaRica node_selected in a pre-defined library. For instance, nodes Dataflow1a and Dataflow1b are instances of the formal node DataFlow that has two inputs I and R and one output O. When it is in its correct mode and its resource is in its correct mode, this node propagates on its output the value of its input, it does not propagate any value if the node or its resource is in the lost mode and otherwise an erroneous value is propagated.

```
node DataFlow
  flow
    O: FailureType:out; I,R: FailureType:in;
  state
    Status: FailureType;
  event
    F_loss,F_error
  trans
    (Status ≠lost) |− F_loss →Status:= lost;
          (Status = ok) |− F_error →Status:= erroneous;
  assert
    O = case {
              Status=ok and R=ok: I,
              Status=lost or R=lost: lost,
              else erroneous
            };
```

```
  init
    Status:= ok;
edon
```

The **state** section declares state variables names and domains. Values of variable Status are in user defined domain FailureType that is the enumeration `ok`, `lost`, `erroneous`. These values denote the failure status of a node. The `correct` value is used when the node is working correctly, the `lost` value is used when the node is not producing any output, the value `erroneous` is used when the node is producing an output whose value deviates from what is expected. The **flow** section declares variables that are used to model data exchanged with interfaced components, the FailureType domain is used.

The **event** section declares failure events. In the **trans** section, a transition is associated with failure event. The transition associated with event `F_loss` may only be triggered when the node is not lost. The new value of `Status` is lost. The value of a state variable can only be modified by transitions.

The **assert** section, defines how the value of output `O` is computed using the values of inputs `I` and `O` and state variable Status.

The **init** section states that initially the component is working correctly.

A library of AltaRica nodes was developed in order to help building safety models for avionics platform architectures. This library includes AltaRica nodes that describe functional, logical and physical nodes. The library was used in order to develop safety models to assess the safety of Integrated Modular Avionics [9]. In this type of architecture, computation or communication resources are shared by several functions or data flows. Consequently, the fault of a shared physical node has an impact on all logical and functional nodes that are connected with this physical node. For instance, in the previous figure, data flows DataFlow1a and DataFlow1b are connected to the same physical node called Phy_Item1. So if Phy_Item1 is lost then both data flows would be lost.

2.2 Safety Analysis

The design of aeronautics safety critical systems deals with two families of faults: random faults of equipments and systematic faults in the development of the equipment, which include errors in the specification, design and coding of hardware and software. Two different approaches are used when assessing whether the risk associated with these two types of faults is acceptable. Qualitative requirements (minimal number of failures leading to a Failure Condition) and quantitative requirements (maximal probability of a Failure Condition occurrence) are associated with equipment faults whereas requirements stated in terms of Development Assurance Levels (DAL) are associated with development faults.

For both types of requirements, the first step of the safety analysis is the computation of the minimal combinations of node faults leading to the Failure Conditions. These combinations are called Minimal Cut Sets (MCS). They are used to compute the mean probability of the Failure Condition in order to assess whether the designed architecture is safe enough. The MCS are also analysed

to check whether there are combinations made of a single fault event that could lead to the Failure Condition.

In order to generate the MCS we used the sequence generator from the Cecilia OCAS toolset. The tool takes as input a Failure Condition (actually it is the name of the output variable of a node representing the Failure Condition in the model) and an Order (the maximum size of the scenarios), then it computes a set of minimal sequences of events in the model that lead from the initial state to a state where the Failure Condition is satisfied. A MCS is a sequence of pairs of the form CpName.FName where, CpName is the name of a node of the AltaRica model and FName is either F_loss or F_error.

Several kind of analysis of the MCS are possible. In the domain of Integrated Modular Avionics (IMA), a first analysis can be performed on the basis of MCS that only include functional node faults. This first analysis provides an indication of the safety of the system before integrating the system on the IMA platform. Then another analysis if performed using MCS that only include physical node faults. This second analysis is used to assess the safety of the system after integration on the IMA platform. The size of both MCS can be compared in order to check whether the effect of common mode failures related wiht IMA shared resources is acceptable.

Another analysis of MCS is performed in order to check the DAL allocation. We check that the DAL allocated to each node dominates the DAL of the Failure Conditions it contributes to. A node contributes to a Failure Condition if a fault of this node appears in one of the MCS of the failure condition. Table 2 gives the basic DAL allocation rules for Failure Conditions (Sev is the severity of the Failure Condition).

Sev	NSE	MIN	MAJ	HAZ	CAT
DAL	E	D	C	B	A

Fig. 2. Basic DAL allocation

New DAL allocation rules introduced in the revised ARP4754a [8] allow to downgrade the original DAL allocated using the basic rule, in cases when nodes involved in the minimal cut sets are known to be pair-wise independent. In order to check these new rules Onera has developed the DALculator [1]. This tool uses as input a file containing MCS for a Failure Condition It also needs an indication of the reference DAL of the Failure Condition. For instance, if the DAL of the Failure Condition is B, and a MCS leading to this Failure Condition is `{Cp1.F_loss, Cp2.F_loss, Cp3.F_error}` then allocating DAL B to Cp1 and DAL D to Cp2 and Cp3 would be acceptable according to the new DAL allocation rules provided that Cp1 is independant from Cp2 and from Cp3. Consequently, it is possible to build highly dependable systems at DAL B with components of mixed DAL B and D.

3 Towards Model Based Security Assessment

3.1 Security Models

We aim at reusing the safety node library in order to build security models. We investigated the main Threat Conditions for the avionics platform and concluded that availability and integrity concepts in safety and security were very similar. Consequently, we propose to use the value `lost` also to denote the status of a component that was subject to an availability threat. We name T_block the generic threat that leads to losing the availability of an item in the architecture. We propose to use the value `erroneous` for the status of a component that was subject to an integrity threat. We name T_forge the related generic threat.

The main difference between Threat Conditions and Failure Conditions is that there is no direct counterpart to confidentiality in the safety domain. We first thought that, as we were interested in "security for safety", we could avoid dealing with confidentiality. But a number of security mechanisms rely on secret attributes as keys or passwords. If the confidentiality of these attributes is compromised then the security mechanism cannot work properly and this could lead to a safety problem. So we decided to deal with confidentiality. We extended the domain FailType with a new value, named `public`, which represents the status of a component whose confidentiality was compromised. We name T_listen the related generic threat.

The platform model was extended with an Agent layer in order to be able to relate a threat with the agent that initiates the attack. The node associated with an agent is very simple, it contains an event T_initiate that represents attack initialisation. Whenever an attack is initiated, the output of the agent node is set to the Boolean value true and this value is propagated to the items located in the zone where the agent is located.

In safety models, the propagation of a failure mode follows the links connecting nodes. We propose to use the same principle in a security model. Let's consider the example shown in the following figure. A threat in physical item Phy_item was exploited by an Agent, its status is erroneous. The picture shows the propagation of this integrity threat from Phy_Item to function Rec_Function. To ease the understanding of propagation, the colours of nodes and links depend of their current value. In the picture, erroneous components and links are coloured in red whereas components coloured in green are working properly. The propagation path starts at component Phy_item, then it goes to Log_item whose integrity is comprised due to its reliance on Phy_item, then Em_Function is also erroneous because it relies on the Log_Item. Then the integrity attack propagates through the functional layer to go from the emitter function to the receiver function.

Other types of propagations should be taken into account when dealing with security: indirect propagation through shared resources. For instance, when a logical item as a piece of software is attacked, if the computer that hosts the software is not protected this attack also contaminates the computer. And then, when the computer is attacked it is also likely that all other pieces of software

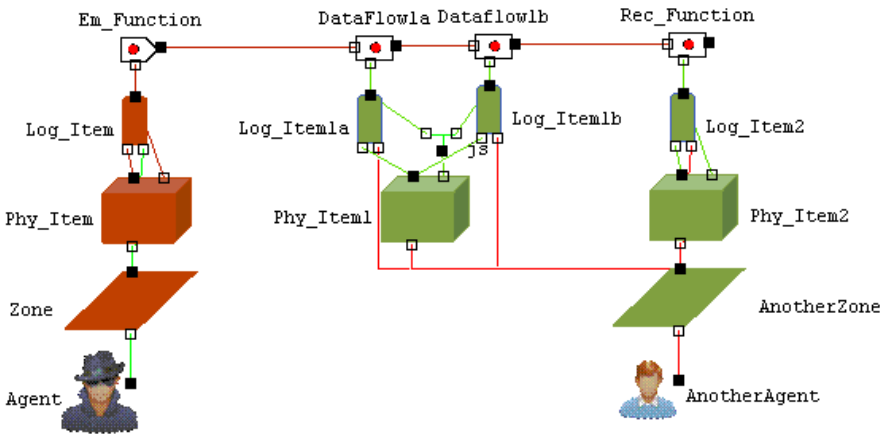

Fig. 3. Propagation of threats through the functional layer

hosted by the computer are contaminated. Similarly, communication links are shared by several computers. It is likely that an attack on one computer can contaminate the communication link and the connected computers. We have extended the models in order to describe this type of indirect propagation. We added event T_contaminate to nodes in the physical layer such that when one of the logical node using this physical node is attacked, the attack is propagated to the physical item and to all the other logical nodes using this physical node.

3.2 Security Mechanism Library

Security mechanisms are nodes that cannot be reused from the safety models. We developed a library of models of security mechanisms that can be used to secure an avionics platform, this library includes models for :

- Zone Access Control: this mechanism blocks the attack initiation signal sent by an agent. This is implemented by controlling the physical access of agents into a zone of the platform. This is an organisational security mechanism.
- Physical Item Access Control: this mechanism blocks the attack initiation signal sent by an agent. This is implemented by controlling the access of agents to a physical item.
- Contamination Control: this mechanism blocks the contamination of a shared resource by an attacked logical item. This can be implemented by an Operation System partitioning service or by virtualization tools.
- Local integrity mechanism: this mechanism blocks an erroneous value and transforms it into a lost value. This could represent a message filtering device.
- End to End integrity mechanism: this mechanism is made of a pair of mechanisms: the first one prepares the proof of integrity of a value and the second one checks the integrity proof. This represents digital signature.

Each security mechanism is implemented as an AltaRica node containing assertions that formalize their influence on the propagation of threat values as (erroneous, lost or public). All the nodes in the security mechanisms library contain an event called T_bypass that voids the limiting effect of the mechanism on threat propagation.

3.3 Data Loading System Model

Using the nodes in the library of layered architectures and in the library of security mechanisms we have developed a model of the Data Loading System. This system is in charge of loading new software releases in the embedded computers of the aircraft.

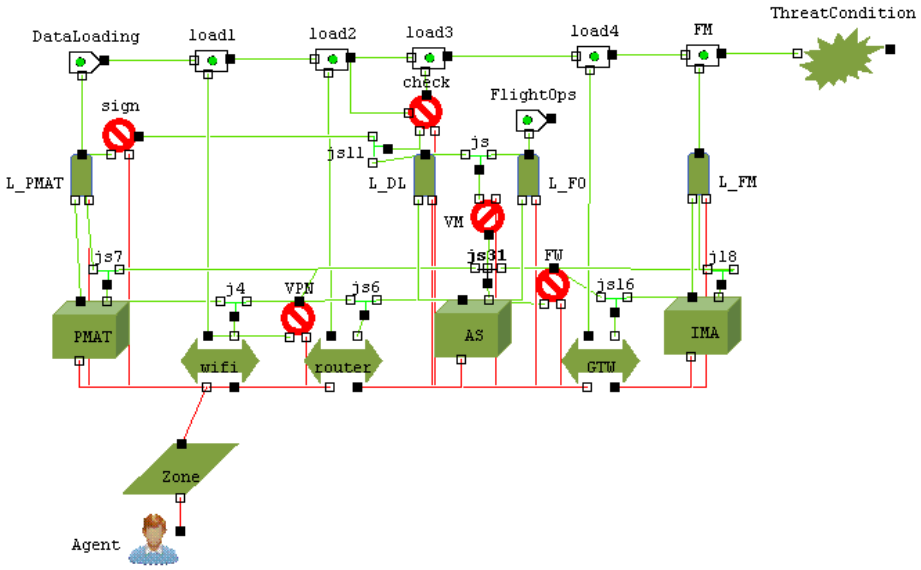

Fig. 4. DLCS Security Model

– Functional Layer: We have detailed the part of Data Loading functions that are related with a request for data loading initiated from a portable maintenance terminal (PMAT) that is connected via wifi to the platform. The load dataflow is emitted by the DataLoading function and is received by the Flight Management (FM) function. The load dataflow is divided into 4 nodes in order to make it easier to relate the data flow with the supporting resources. The functional layer also contains a very abstract view of the flight operation function (FlightOps). This function is described by a unique node. In this model, its role is to study the potential contamination between various functions.

- Logical Layer: the model consists of 4 logical nodes to describe the software components needed to implement the Data Loading (and part of the Flight Ops) functions. For the sake of simplicity, we have not included logical nodes for the network components (wifi, router, and GTW). In that case, data flow nodes are directly linked to physical resources
- Physical Layer: the model includes the major physical nodes useful for the Data Loading functional chain: computers such as PMAT, AS (Avionics Server) and IMA (Integrated Modular Avionics) and communication equipment such as wifi, router and GTW (Communication Gateway).
- Zone Layer: The model contains only one zone of interest where the wifi link can be attacked. We have not modelled all the other zones of the aircraft because we have supposed that either the agents that can access the platform are trusted (pilot, cabin crew, ..) or there are sufficient organizational controls in order to stop an attack that would originate in other zones such as the cockpit, cabin or aircraft vicinity.
- Agent Layer: the model contains only one type of agent that represents the General population as we have considered, for this example, that all other types of agents are trusted.

Two kinds of security mechanisms were used in the model:

- End to End Integrity Control is used to model the signature of loads with a digital signature exchanged by the PMAT (sign component) and the AS (check component).
- Contamination Control is used three times. VM models a virtualisation service running on the AS server in order to control the contamination of logical items. VPN models a virtual private network mechanism associated with the router, FW models a firewall associated with the gateway.

3.4 Security Analysis

We reuse safety analysis tools that were presented earlier to assess the security of the model. We use the Cecilia OCAS Sequence Generator in order to generate threat scenarios leading to a threat condition. We want to use the DALculator in order to analyse threat scenarios in order to allocate a security level with nodes of an avionics platform.

We were interested in two Threat Conditions related with the Data Loading system:

- *"Loss of update of Flight Management software due to a Data Loading via wifi"*. We consider that this Threat Condition is moderately severe because this could potentially lead to the loss of the Flight Management system when the aircraft is on ground.
- *"Erroneous update of the Flight Management software due to a Data Loading via wifi"*. We consider that this Threat Condition is more severe because this could potentially lead to an erroneous behaviour of the Flight Management system during flight.

We generated all threat scenarios including a maximum of 6 threat events. As in the case of MCS, a threat scenario is a sequence of pairs of the form CpName.ThreatName that starts from the initial state of the system where no attack is launched and leads to a state where the system is attacked. In a threat scenario, CpName is the name of a logical node or a physical node or a security mechanism or a user name, and ThreatName is the name of the threat being activated on the component. It could be: T_forge (corruption threat), T_listen (disclosure threat), T_block (denial of service threat), init (user attack activation) , T_bypass (circumvention of a security mechanism) or T_contamination (propagation of a threat from one component to another).

Let us consider a threat scenario leading to the erroneous update of the Flight Management software.

{'Agent.T_init', 'PMAT.T_forge'}

This Threat Scenario is made of two pairs. The first pair 'Agent.T_init' means that the user called Agent is the initiator of the attack and the second pair 'PMAT.T_forge' means that a corruption attack is performed on physical node PMAT. The corruption of the physical node has a negative effect on the behaviour of the end to end integrity protection mechanisms. As this could lead to an undetected corruption of the load then it could be the case that the flight management function is updated with a corrupted load.

Let us now consider a more complex Threat Scenario leading to the loss of update of the Flight Management function. This scenario involves the propagation by the router of an attack.

{'Agent.T_init', 'router.T_contamination', 'AS.T_block', 'router.T_contamination'}

In this scenario, Agent initiates the attack. The next step is 'router.T_contamination'. This means that the router could propagate the attack initiation signal to components physically linked with the router (e.g. wifi). The following step is 'AS.T_block', which means that a denial of service attack is performed on the avionics server. The last step of the scenario is 'router.T_contamination', which means that the denial of service attack is propagated to the router and all components connected to the router. Since the router is lost (due to the propagated denial of service attack) and should be used to communicate the software load, then the software load is not received by the Flight Management function that cannot be updated.

Let us now consider another threat scenario:

{'Agent.T_init', 'router.T_contamination', 'AS.T_block', 'FW.T_bypass', 'GTW.T_contamination'}

In this scenario, Agent initiates the attack, and then the router propagates the attack initiation signal to nodes physically linked with the router. Then, a denial of service attack is performed on the avionics server. The next step of the scenario is 'FW.T_bypass', which means that the firewall is deactivated. In the last step the denial of service attack is propagated to the gateway.

Among the 13 threat scenarios that lead to the loss of the Flight Management there are 11 scenarios of size 2, and 2 of size 3. Among the 40 threat scenarios that

lead to an erroneous behaviour of the Flight Management there are 9 scenarios of size 2, 21 of size 3 and 10 of size 4. All these scenarios can be reviewed in order to check whether there are enough security mechanisms in the avionics platform.

We used the DALculator tool to check the allocation of a Security Level to each of the components that appear in the threat scenarios. The Security Level of security mechanisms measures the expected efficiency of this mechanism. The Security Level for other nodes can be seen as a level of Trust in the node. The Security Level is measured using the same range of values as the DAL : from E to A.

Due to the severity of the Threat Condition *"erroneous update of the FM software due to the DLCS via wifi"* we have considered that its Security Level is B.

We have considered that the Security Level of nodes Agent and wifi is level D. This means that their trust level is very low. We considered that the Security Level of the node Flight Ops is level C because it is little bit more trusted than external entities as the Agent or the wifi network.

The solution checked by the DALculator is described in the following figure. In this solution the efficiency of the end to end integrity functions (sign and check nodes) should be rather high (level B). The trust in the resources used to check the signature is consistent as AS and L_DL should also have level B. A similar efficiency is required for VPN and VM. The level of trust in other components is rather low (level D).

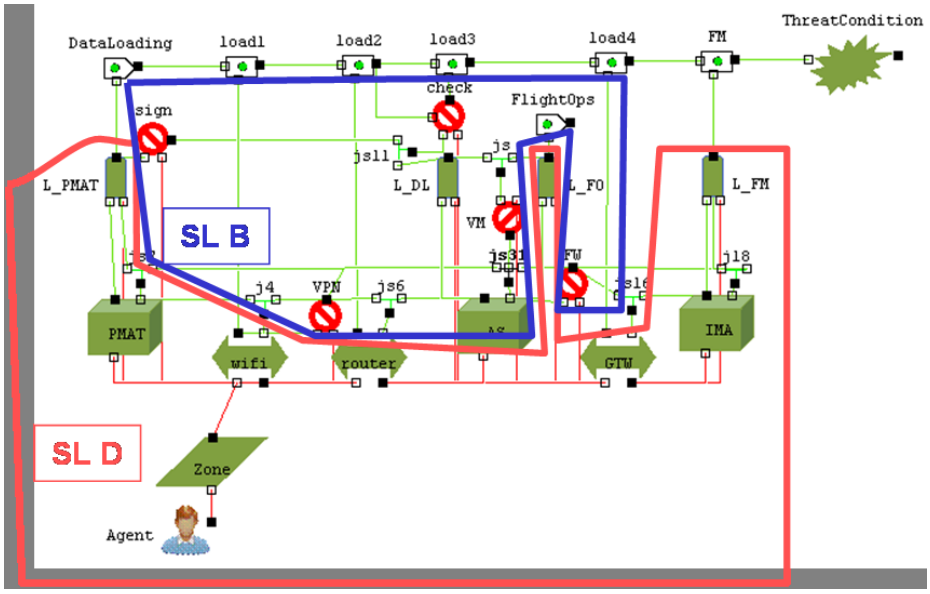

Fig. 5. Security Level Allocation

4 Conclusion

4.1 Preliminary Lessons Learnt

This first experiment in using AltaRica to build security models helped us to analyse convergence and divergences points between Safety and Security modelling and assessment:

- The layered model for the safety analysis of avionics platform was reused efficiently for the security model. We only had to add the Agent layer to model threat activation.
- The AltaRica code was easily extended to deal with security threats. The main addition was related with the modelling of confidentiality.
- We had to model the propagation of threats due to the use of shared resources. Although, propagation of an integrity or availability fault is a safety concern that is considered in IMA developement we never included this type of propagation in the safety models we have been building so far in order to help System safety Assessment.
- The OCAS sequence generator was used successfully to generate Threat Scenarios. As the models could include both fault and threat events, it should be possible to generate scenarios that combine fault and threats in order to analyse complex requirements mixing safety and security.
- Using DAL to model the security level of nodes in the architecture was possible. Extra work is needed in order to check the consistency of DAL allocation rules with security level allocation rules such as the ISO 27005 risk reduction approach (see [3] for a comparison of DAL level and Security Level).

4.2 Related Work

The work described in this paper is preliminary. The most relevant related work seems to be the attack tree approach [5]. This approach proposes to use the classical fault tree notation in order to study security. As in our work, the attack tree contains basic events representing elementary threats. In some variants of the notation, the tree also include a description of the effect security barriers. In [7] the authors propose to use an extension of the fault-tree notation in order to deal with dynamic aspects of the threat propagation. Both of the previous works tend to focus on a quantitative assessment of security requirements whereas we have been working on qualitative requirements because this would be more consistent with the Airworthiness Safety process. Another relevant approach was proposed by the CORAS project [6] , this notation aims at assisting the security risk analysis. A difference between this approach and our work is that the CORAS can be applied before the security architecture is designed whereas our approach is applied once the security architecture is established.

Acknowledgements. The research described by this paper was partially supported by the ITEA project MERGE and DGAC project ARCS.

References

1. P. Bieber, R. Delmas, and C. Seguin. Dalculus – theory and tool for development assurance level allocation. In *Proceedings of the 30th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2011),*, Lecture Notes in Computer Science. Springer-Verlag, 2011.
2. P. Bieber and C. Seguin. *Industrial Use of Formal Methods: Formal Verification*, chapter chapter 3: Safety Analysis of Embedded Systems with the AltaRica Approach. Wiley, 2013.
3. J.-P. Blanquart, P. Bieber, G. Descargues, E. Hazane, M. Julien, and L. Leonardon. Similarities and dissimilarities between safety levels and security levels. In *Proceedings of the Embedded Real-Time Systems and Software Conference (ERTS2 2012)*, 2012.
4. M. Bozzano, A. Villafiorita, O. Aakerlund, P. Bieber, C. Bougnol, E. Böde, M. Bretschneider, A. Cavallo, C. Castel, M. Cifaldi, A. Cimatti, A. Griffault, C. Kehren, B. Lawrence, A. Luedtke, S. Metge, C. Papadopoulos, R. Passarello, T. Peikenkamp, P. Persson, C. Seguin, L. Trotta, L. Valacca, and G. Zacco. Esacs: an integrated methodology for design and safety analysis of complex systems. In *Proceedings of ESREL 2003*. Balkema publisher, 2003.
5. B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer. Attack-defense trees. *Journal of Logic and Computation*, 24:55–87, 2012.
6. M. S. Lund, B. Solhaug, and K. Stoelen. *Model-Driven Risk Analysis. The CORAS Approach*. Springer, 2010.
7. L. Piètre-Cambacédès and M. Bouissou. The promising potential of the bdmp formalism for security modelling. In *Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, 2009.
8. S. S-18 and E. W.-. committees. Arp4754a - guidelines for development of civil aircraft and systems. SAE aerospace, 2010.
9. L. SAGASPE, G. BEL, P. BIEBER, F. BONIOL, and C. CASTEL. Safe allocation of shared avionics resources. In *Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE 2005)*, 2005.
10. WG72. Ed202 - airworthiness security process specification. EUROCAE, October 2010.