ITEA2 – Project #11011

Multi-Concerns Interactions System Engineering

01.12.2012 to 31.03.2016

Recommendations for Security and Safety Co-engineering (Release n°3)

Project Deliverable D3.4.4 - Part B (Julien Brunel, ONERA)

Task 3.4 – Advanced concepts in safety and security co-engineering (Laurent RIOUX, TRT)

WP3 – Advanced multi-concerns engineering concepts (Sam MICHIELS, KUL)

Status	[ ] Draft	**Document Created**	: 18.08.2015
	[ ] To be reviewed	**Last edited**	: 22.04.2016
	[X] Final	**Due date**	: 28.02.2016
Confidentiality	[X] Public (for public distribution)	**Ready for review**	:01.02.2016
	[ ] Restricted (only MERgE internal use)	**Document Version**	: 1.0
	[ ] Confidential (only for individual partner(s))	**Pages**	: 92

Contributors : TRT, ALL4TEC, ONERA, STUK

Executive summary

Nowadays, safety and security are two risk-driven activities that are tackled separately, giving rise to the industrial challenge of efficiently and economically co-engineering these two specialities. It is evident that there is a major opportunity to share on onomastics[1], algorithms, (formal) methods and tools, in particular to reach higher levels of assurance at contained costs.

Deliverable D3.4.4 is split in two parts. Part A (companion document) is an extensive state of the art on safety and security co-engineering of software intensive critical information systems. It essentially covers academic publications and industry standards.

Part B (this document) first reports on two prototype tools dedicated to safety and security co-engineering. The first prototype was designed and developed by MERgE partners based on safety and security requirements from the MERgE software-defined radio test case. The document recalls the requirements and presents the high-level design. Assessment results of this prototype can be found in deliverable D1.1.1d – TCS Evaluation. The second prototype, called AVATAR, is developed by Télécom ParisTech and was identified during our study of the state of the art. We performed an in-depth assessment of this academic tool. Based on the experience we gained during the state of the art work (of which a synthesis is provided herein) and tool prototyping work, Part B proceeds with research and development recommendations for new federative approaches, whilst remaining realistic with respect to industrial constraints, i.e. costs, legacy workbenches, training constraints, etc.

Note: the executive summary is common to both parts A and B.

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

[1] Study of names and naming.

Version	**Content**	**Resp. Partner**	**Date**
0.01	Creation of the document based on D3.4.3. Enhancement of the state of the art.	S. Paul (TRT) L. Rioux (TRT) J. de Oliveira (TRT) G. Gailliard (TCS) J.-L. Gilbert (TCS) T. Wiander (STUK) F. Vallée (All4Tec) M. Bakkali (All4tec) A. Faucogney (All4tec) J. Brunel (ONERA) D. Chemouil (ONERA)	18/08/2015
0.02	Extension of the state of the art with the papers from the ISSE'15 workshop.	Stéphane Paul (TRT)	22/09/2015
0.03	Further enhancement of the state of the art.	Stéphane Paul (TRT)	06/10/2015
0.04	Split of the document in two parts. Contribution on TTool/AVATAR.	Stéphane Paul (TRT)	24/11/2015
0.05	Correction of formatting errors.	Stéphane Paul (TRT)	30/11/2015
0.06	Adapted Executive Summary and Introduction. Restructured chapters 2 and 6. Solved chapter reference errors, solved printing problems and other minor issues.	Stéphane Paul (TRT)	04/01/2016
0.07	Correction of the document template. Restructuring of §3. Restructured and extended §6.	S. Paul (TRT) L. Rioux (TRT) F. Vallée (All4Tec) M. Bakkali (All4tec) J. Brunel (ONERA)	07/01/2016
0.08	Extended state of the art. Clean-up of bibliography (non-cited references).	Stéphane Paul (TRT)	12/01/2016
0.09	New section §3.2, and adaptation of §3 and §3.1. Update of Fig. 1.	Julien Brunel (ONERA) Stéphane Paul (TRT)	19/01/2016
0.10	Revised §3, new section §3.3. Submission for internal review.	Stéphane Paul (TRT)	25/01/2016
0.99	Peer review.	Grégory Gailliard (TCS)	16/02/2016
1.00	Final submission.	Stéphane Paul (TRT)	16/02/2016

Reviewed & Accepted	**Name**	**Partner**
Independent Reviewer (outside WP)	G. Gailliard	Thales Communications & Security
Validation from Management Board	S. Michiels	Katholieke Universiteit Leuven

Contents

List of Figures

There will always be engineering failures. But the worst kinds of failures are those that could readily be prevented if only people stayed alert and took reasonable precautions. Engineers, being human, are susceptible to the drowsiness that comes in the absence of crisis. Perhaps one characteristic of a professional is the ability and willingness to stay alert while others doze. Engineering responsibility should not require the stimulation that comes in the wake of catastrophe.

—Samuel C. Florman
The Civilized Engineer

Extended executive summary

Safety and security are two risk-driven activities that are traditionally tackled separately. It is thus possible to distinguish two communities, each working on their own standards, organising their own conferences, publishing in their own journals. Since the 9/11 attacks on the Twin Towers in the Aeronautics domain and the discovery of the Stuxnet computer worm in the Industrial Control Systems domain in June 2010 (cf. Figure 1), it is more and more recognised worldwide that both engineering specialties cannot continue to ignore each other.

Early 2014, when we started this state of the art on safety and security co-engineering for software-intensive systems, we thought we would rapidly establish a comprehensive picture of this small community living in the shadows of the big safety community on the one hand, and of the security community on the other hand. In our minds, safety and security co-engineering questions where intimately linked to niche safety-critical systems markets, such as the Integrated Modular Avionics (IMA), Industrial Control Systems (ICS) or similar networked control systems.

Much to our surprise, we discovered a bustling academic community, with a significant number of publications explicitly addressing safety and security co-engineering concerns (cf. part A, §2), and actively organising workshops and conferences on the subject (cf. part A, §6). As illustrated in Figure 1, our state of the art on academic safety and security co-engineering publications comprehends some 160 references (on a total of over 400 references in the deliverable) concentrating essentially on the last 10 years[2], even if a few references go back to the early 90's. Recent attention to the topic may be related to the explosion of the number of cyber-physical systems (CPS), system of systems (SoS) and Internet of Things (IoT) in general public markets. We also found an industrial community actively revising existing safety-related standards or elaborating new standards to cope with business security issues with a certain level of rigor (cf. part A, §3). This standardisation activity is all the more surprising that there is a real lack of international regulation concerning security risk management for safety-critical systems. The last but not least of our surprises was in the education domain: there seems to be very few courses addressing both cyber-security and safety engineering, which does not bode well for the future (cf. part A, §5).

Our state of the art was first organised in a chronological order (cf. part A of this deliverable), and then analysed as a whole. This analysis led us to organise the publications in three groups (cf. part B, §5). A first group comprehends the papers that state the issues related to engineering safety and security separately, and assert that there is room for improvement, but do not explain how. The second group comprehends the papers that propose to improve one specialty by adapting techniques from the other specialty, in other words, safety and security cross-fertilisation. Here, one specialty is seen as more important than the other one, giving way to *security for safety* or vice-versa.

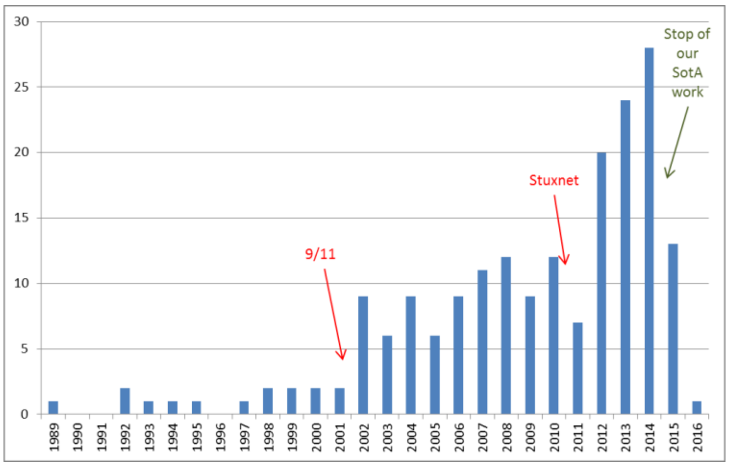


Figure 1: Number of safety and security co-engineering related research publications per year

The last set of publications relates to novel clean-slate approaches for safety and security co-engineering, considering both specialties as peers. Amongst these publications, one tool, called TTool/AVATAR, caught our attention and was analysed in depth (cf. Part B, §4).

From the mass of aforementioned publications and after an analysis of internal MERgE case test safety and security co-engineering requirements (cf. part B, §2):

- we identified and developed a new formal system modelling and verification framework for security and safety assessment (cf. part B, §3), which extends the classical safety-related dysfunctional modelling with security-related concerns;
- we ventured to formulate a couple of facts, and a couple of trends (cf. part B, §6).

[2] The number of references for 2015 is significantly low due to the fact that we stopped our systematic search of publications early 2015, and simply referenced occasional findings.

The first fact is that safety and security co-engineering seems to be primarily a concern of the safety engineering community. Indeed, the increasing number of cyber-attacks in the world tends to show that safety-critical systems, and in particular the rising number of cyber-physical systems, which are particularly exposed by nature, may not be as safe as they claim, if they are not also secure. The multiplication of security-related workshops in conjunction to safety-related conferences, and the multiplication of safety standards updates that include security concerns both provide significant testimonies of this growing interest for safety and security co-engineering by the safety community. There is no similar booing within the security community: security experts seem to be interested in safety studies in only two cases: (i) to assess if safety-critical systems are more vulnerable when they switch into fail-safe modes; (ii) to re-use safety techniques when availability and integrity are the primary concerns of the security engineering work, by opposition to confidentiality or privacy concerns.

The second major fact is that the security regulation is somehow lagging behind industrial initiatives to produce security standards. This may be explained considering that security is a National sovereignty prerogative, whilst safety regulation has often been transferred to transnational organisations (e.g. European Commission, International Civil Aviation Organisation) since decades. Depending on the domains, National regulation may be seen as too weak or on the contrary an effective means to affect worldwide businesses. In the nuclear domain, renewed national regulation is a driver for unified safety and security considerations as the example of STUK YVL guides suggest. Other industries (e.g. in the aviation domain) have been developing security standards, which cannot be termed as acceptable means of compliance (AMC), since there is no regulation to comply with. This situation is bound to change.

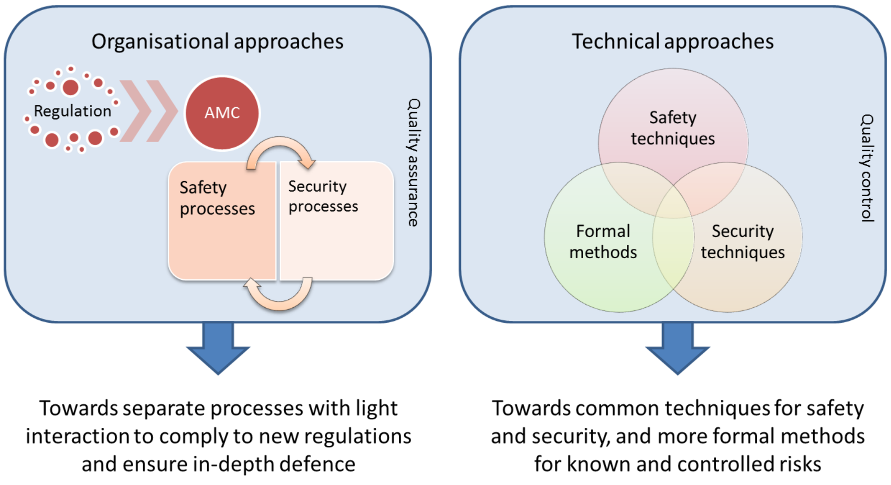

Organisational approaches
Regulation
AMC
Safety processes
Security processes
Quality assurance
Technical approaches
Safety techniques
Formal methods
Security techniques
Quality control
Towards separate processes with light interaction to comply to new regulations and ensure in-depth defence
Towards common techniques for safety and security, and more formal methods for known and controlled risks

Figure 2: Identified trends in safety and security engineering

Trends (cf. Figure 2) were a bit more difficult to establish. We have formulated two of them based on concordant events happening in multiple domains (e.g. aviation, electronics, nuclear), and on both side of the Atlantic:

- the safety communities thrive to maintain current organizational approaches as stable as possible, because regulations, acceptable means of compliance and standards have proven efficiency records and are extremely difficult to change, technically and / or politically; some minor updates to the processes and methods are however necessary to ensure interaction points, such as safety-aware security in the avionics domain, or security-aware safety in the electrical / electronic / programmable electronic domain; the safety communities seems to be moving away from revolutionising standard safety processes, even if all individual members of those communities do not seem to adhere to this trend;
- the academic and industrial communities are adapting and extending existing, architectures and tools, to cover both safety and security properties; within this trend, the adoption and seamless integration of formal methods and tools occupies a significant part.

These two trends cover quality assurance for the former, to ensure in-depth defence, and quality control for the latter, to cope with known and controlled risks. All of the above is detailed in the current document.

The document concludes on a set of proposals for continued enhanced safety and security co-engineering. The proposals are based on a set of three assumptions:

- industrial safety and security engineering processes / methods are difficult / slow to change;
- safety and security vernacular is difficult / slow to change;
- safety and security tools are diverse, but tend towards a formalisation of their conceptual data model.

Based on these hypotheses, our state of the art and our technical work, the document concludes on three proposals that may feed a safety and security co-engineering research and development roadmap:

- the development of a common pivot model to support artefact sharing between engineering specialties;
- the management of conflicts between safety and security engineering processes seen as independent processes;
- the criteria to be respected by engineering tools to allow for successful cross-fertilisation between engineering domains.

1 Introduction

1.1 Purpose of the document

The purpose of deliverable D3.4.4 is to provide research and development recommendations for safety and security co-engineering of software-intensive safety-critical information and / or embedded systems. The deliverable is split in two parts. This document is part B.

The recommendations (cf. §6) are based on a solid state of the art (cf. Part A of this deliverable), of which a synthesis is provided herein (cf. §5). The recommendations are also backed by end-user requirements from the MERgE end-users (cf. §2), and on technical work related to two prototypes: an internally developed tool (cf. §3) and a third-party tool (cf. §4).

1.2 Scope of the document

The scope of the study is the following:

- focus on the safety and security co-engineering of **software-intensive critical information and / or embedded systems**, but not excluding other systems;
- **end-to-end** safety and security **co-engineering**, i.e. from safety and security requirements elicitation, through to the implementation of safety and security solutions, and the verification and validation of those properties;
- safety and security co-engineering **modelling** methods and tools.

This document addresses neither safety and security taken independently, nor safe and secure computing solutions which do not require engineering practices.

1.3 Motivations

There are at least five main motivations for driving this study about safety and security co-engineering.

Motivation n°1: the question is no more *if* your system is going to be subject to a cyber-attack, but *when*.

On Dec. 8th, 2014, the SC Magazine (Stephenson, 2014) makes a title on *Information security in 2014: another year of big events*, and the article starts as follows: *As 2014 draws to a close we can look back over one of the most tumultuous years in recent history. This has been the year of the major security breach. The Target breach was just a warm-up for a laundry list of attacks against large, presumably well-protected, companies and government agencies. Candidly, these organizations – public and private – should be ashamed of themselves*. Undeniably, from the cyber-attack point of view, the world is becoming more dangerous every day. As end-users awareness increase, they now consider normal that up to 6% of a safety-critical system's cost may be dedicated to security issues.

Motivation n°2: safety-critical systems are no more an exception to the rule, being them also subject to cyber-attacks.

As safety-critical systems become more and more complex (cf. Figure 3), and more and more interconnected, cf. (25-356-SC, 2008) and (25-357-SC, 2007), they also become more and more vulnerable to cyber-attacks. A major driver of this evolution is the increasing number of software updates, versus hardware upgrades. This requires ports and protocols for remote maintenance / configuration, which are as many openings for malevolent actions.

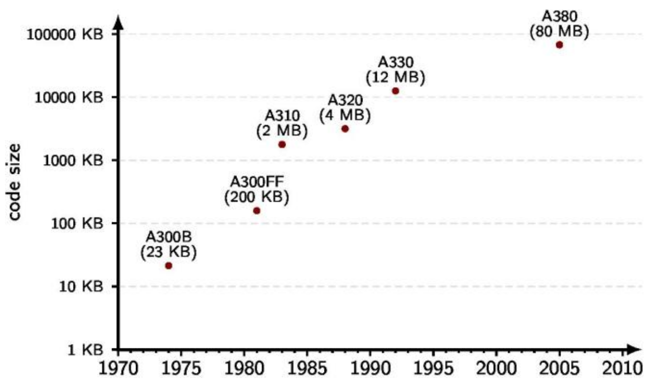


Figure 3: Exponential code size evolution on Airbus aircraft

Motivation n°3: components-off-the-shelf (COTS) have become ubiquitous in software engineering.

The Service and Component Architecture (SCA) has effectively leveraged reuse in the software engineering process, including for safety-critical and/or embedded systems. However, when COTS are massively used in the software design, proving overall safety and security properties is a real challenge. A strategy for COTS selection needs to be defined beforehand, together with guidance on how to use / configure them.

Note: sub-contracted software development fall under the same category as COTS when limited trust is granted to the sub-contractors with respect to the existence of backdoors and / or Trojan horses in the delivered software.

Motivation n°4: system maintenance in secure conditions (MSC) and system maintenance in operational conditions (MCO) go by very different update rates and live cycles.

Safety-critical systems are hard to certify; once certified, modifications are kept minimal in order to avoid running the complete certification process all over again. On the contrary, system maintenance in secure conditions requires frequent updates to keep up with the ever rising new threats and related patches. Living with both these safety and security constraints requires well-thought system architectures.

Motivation n°5: there are no complete and convincing solutions on the market to address simultaneously safety and security engineering, including trade-off decision support.

The safety engineering has a long history of good practices, standards and tools, which have reached a high degree of maturity. The security engineering domain is newer and is subject to constant evolution. Both communities have lived side-by-side with few interactions. One partial exception to this statement is the MILS architecture used for real-time operating systems (RTOS). The MILS architecture assures properties that are relevant to both safety and security, typically *non-bypassable*, *evaluable*, *always invoked*, and *tamperproof*. MILS currently appears in commercial products, e.g. PikeOs by (Sysgo, 2014), Integrity Multivisor by (Green Hills Software, 2014), VxWorks by (Wind River, 2015), LynxOS by (Lynx Software Technologies, 2015)[3] or QNX® Hypervisor by (QNX, 2015). However, by itself, MILS is far from being a complete solution, to cover the complete safety lifecycle, from the functional hazard analyses and safety cases, to verification and validation.

1.4 Targeted audience

The targeted audience of this release of the deliverable is the safety and security co-engineering community, without any restriction.

1.5 Structure of the document

This document is structured as follows.

- Chapter 1 is the current introduction.
- Chapter 2 recalls our end-user safety and security co-engineering requirements, as expressed in the MERgE test cases.
- In chapter 3 and 4 we report on two formal safety and security verification experiments, one with a tool developed in the scope of the MERgE project, one with a third-party tool.
- Chapter 5 provides an overall synthesis of the state of the art (as presented in Part A of this deliverable).
- Based on this synthesis and the above technical work, chapter 6 provides some elements that may feed a safety and security co-engineering research and development roadmap.
- Chapters 7 and 8 provide respectively the references and definitions of acronyms.

[3] Previously known as "LynuxWorks".

2 Analysis of MERgE case test safety and security co-engineering requirements

2.1 SDR safety and security co-engineering requirements

This section recalls the safety and security requirements of the TCS Software Defined Radio (SDR) test case, as extracted from MERgE deliverable D1.1.1.a, *TCS - Scenarios and use cases definition*.

In general, radio communication equipment may have to be compliant with the (ISO/IEC 15408-1, 2009) and (RTCA DO-178B, 1992) / (EUROCAE ED-12B, 1992) standards presented in the next paragraphs. Since radio equipment is long-lived, RTCA DO-178B is still being used. In the long term, (RTCA DO-178C, 2011) / (EUROCAE ED-12C, 2012) should be used. Critical Software Defined Radio (SDR) systems may be designed according to the security architectural pattern defined by Multiple Independent Levels of Security/Safety (MILS). MILS is a security architecture that uses partitioning techniques to host applications with different security levels.

The TCS SDR test case targets an EAL3+ level. The current Design Assurance Level (DAL) required for TCS radio equipment is DAL D. Thus, TCS has identified the following safety and security requirements.

Note: The traceability of the SDR requirements to the test case realisation with the MERgE platform is provided in deliverable D1.1.1.d - *TCS – Evaluation*.

2.1.1 Security requirements

The security viewpoint **should** support risk analyses.

For instance, an attack tree analysis may be handled in the security viewpoint or delegated to an external security analysis tool.

The security viewpoint **should** support security evaluations according to the Common Criteria.

For instance, the security viewpoint should support the concepts of the Common Criteria such as EAL, Threats, TOE Security Functions (TSF), TSF Interface (TSFI) to model the security target.

The security viewpoint **shall** support the definition of security components with their interfaces, parameters and errors messages.

The security viewpoint **shall** support the assembly of security components to specify the security architecture.

At system level, the security viewpoint **shall** support the deployment of waveform and platform components to allowed security domains.

For instance, waveform components are designed to work in the red or black domains.

At software level, MyCCM and SCA component frameworks **shall** verify that waveform components are deployed on the required security domain.

At system level, the security viewpoint **shall** support the specification of allowed connections between waveform components and radio platform components.

At software level, MyCCM and SCA component frameworks **shall** verify that the type of waveform and type of platform component ports are compatible.

At system level, the security viewpoint **shall** support the specification of allowed messages between components, which may depend on a combination of parameters such as:

- Platform state, e.g. boot, initialisation, update, deployment;
- User roles/rights, e.g. end-users, maintainers, security officers;
- Security domains, e.g. red/black;
- Connection between component ports;
- Parameter identifier and value.

At system level, the security viewpoint **should** support the specification of formal rules using a dedicated language to check the validity of requests to services.

At software level, MyCCM and SCA component frameworks **should** perform access control at runtime based on formal rules defined in the system model.

At system level, the security viewpoint **should** support the specification of allowed read, write, read-only access to waveform and platform component properties.

At software level, MyCCM and SCA component frameworks **should** perform access control at runtime based on the read, write, read-only access rights defined in the system model.

At system level, the security viewpoint **should** support the specification of valid ranges for waveform and platform property values.

At software level, MyCCM and SCA component frameworks **should** verify at runtime the range of waveform and platform property values based on the rules defined in the system model.

At system level, the security viewpoint tool **shall** allow the definition of Explicit Secure Communication Channels to define allowed information flows between components.

At system level, the security viewpoint tool **shall** allow the definition of Logical Security Partitions to separate application and platform components with different security requirements.

For instance, the Logical Security Partitions may enforce the separation of waveform applications from the radio platform.

At system level, the security viewpoint tool **shall** allow the definition of Logical Security Partitions to separate different applications from one another with different security requirements.

For instance, the Logical Security Partitions may enforce the separation between waveform applications.

At software level, Security Partitions **should** be defined for the deployment of waveform and platform components on Red and Black security domains.

At system level, the security viewpoint **shall** support the identification of confidential information and the conditions under which they may be accessed.

For instance, confidential information may designate waveform binary code or configuration files stored in the platform file system.

At system level, the security viewpoint **shall** support the identification of information whose integrity must be verified.

At system level, the security viewpoint **shall** support the identification of information whose authenticity must be verified.

At software level, MyCCM and SCA component frameworks **may** verify the integrity and authenticity of waveform and platform components before their deployment on the radio platforms according to system model requirements.

2.1.2 Safety Requirements

The safety viewpoint **should** support safety analysis.

For instance, a fault tree analysis may be handled in the safety viewpoint or delegated to an external tool.

At system level, the safety viewpoint **shall** support the definition of Logical Safe Partitions to specify the isolated execution of waveform and platform components.

At software level, Logical Safe Partitions **may** rely on software partitioning technologies such as software component containers, separation kernels and hypervisors.

At system level, the safety viewpoint **shall** allow the definition of platform properties to be monitored.

Examples of platform properties that may be monitored include CPU, memory and battery consumption.

The safety viewpoint **should** support safety evaluations according to the (RTCA DO-178B, 1992) / (EUROCAE ED-12B, 1992).

2.2 ICS safety and security co-engineering requirements

This section recalls the safety and security requirements identified in the ICS use-case, as extracted from the MERgE deliverable D1.1.4.a: *ICS – Scenarios and use case definition*. There are several public information security standards and frameworks available to help organisations to address ICS security concerns. The following have been selected for the test case:

- NIST 800 series: (NIST SP 800-53, 2013), (NIST SP 800-82, 2013);
- IEC 62443[4] (formerly ISA-99);
- ISA Secure certification for embedded ICS devices.

The traditional way of defining security is to divide it into three sub-requirements namely confidentiality, integrity and availability (CIA). The CIA definition/division applies also in ICS, but the prioritization is different. Whereas confidentiality of data is often the most important requirement in standard ICT systems, the data in ICS is often

[4] See (IEC/TS 62443-1-1, 2009), (IEC 62443-2-1, 2010), (IEC/TR 62443-3-1, 2009), and (IEC 62443-3-3, 2013).

not confidentiality-critical. Integrity of data is equally important in both ICT and ICS domains, especially in industrial automation where lacking integrity in the communication channels can result in significant safety hazards. Availability is often not so critical in ICT system, although downtime in critical business systems can result in loss of business or business continuity.

In ICS on the other hand, availability is the most critical subcomponent of safety and security. The availability requirement in ICS is often solved by separating the communication system from the control system. Even if the system responsible for communications crashes, or reboots, or becomes deadlocked, the control system must be able to continue or must be able to shut down the critical control logic into a safe state. Quite often the digital communications logic is complemented with analogue controls that allow process continuity even in cases of break-down of communications network or components. Shutting down the digital controls can sometimes also be seen as a defence against attacks.

The following list provides examples of design requirements for ICS systems.

Systems **shall** be designed with the assumption of physical security.

No tampering with the devices, software or networks **shall** be allowed to anyone without specific clearance.

System **shall** remain up and running at all times.

There **shall** be no possibility for reboots or reinstalls or patching of security issues.

Scheduled maintenance breaks **shall** happen very rarely, such as once a year or even more infrequently.

Data integrity **shall** be maintained.

Systems **shall** be robust to handle anything thrown at them.

Many ICS communication technologies do not implement any unnecessary security functions. Authentication or data integrity checks are often forbidden as they can result in critical messages being dropped from the systems.

Extensive robustness testing is required by industry standard to show resistance to broken or hostile communications and measurement data.

It should be noted that industry specific standards and frameworks are in development. As example, in the nuclear sector there is a new standard, namely Nuclear Power Plants - Instrumentation and control systems - Requirements for security programmes for computer-based systems (IEC 62645, 2014).

2.3 Other test cases

The two other MERgE test cases are the automotive and aerospace test cases.

The MERgE automotive case study is centred on the Hall Effect Sensors of Melexis. In deliverable D1.1.2a: *Automotive case: Scenarios and use case definition*, only safety and availability requirements have been identified for this case study, i.e. no security requirements, no joint safety and security design process.

Likewise, in deliverable D1.1.3.a, *SASNV - Scenarios and Use Cases Definition*, only safety and dependability requirements have been identified for this test case, i.e. no security requirements.

Thus, for the automotive and aerospace test cases, there are no domain-specific challenges to be considered in this study.

3 A formal system modelling and verification framework for security and safety assessment

In a first section, we present a stand-alone framework, called Coy, that allows:

- modelling system architectures including potential failures of components, security threats, and the propagation of components failures and threats along the architecture, in the spirit of Model Based Safety & Security Assessment (MBS&SA) ;
- checking safety and security requirements, relying on the formal semantics of Coy and the Alloy Analyzer tool.

In a second section, we present an integrated approach that maximises the use of the MERgE platform. In particular, Capella system engineering models can be used as input to the safety and security analyses, without having to redefine the complete system architecture, as with the stand-alone Coy framework.

3.1 Standalone approach

Previous experiments of formal safety and security verification have already been published in (Brunel, et al., 2014a), (Bieber, et al., 2014) and (Brunel, et al., 2014b)[5]. Following these earlier works, we propose here to address the question of safety and security assessment using the Alloy language and the Alloy Analyzer free-software tool. Alloy (Jackson, 2006) is a formal modelling language amenable to automatic analyses.

Our motivation for relying on Alloy instead of, say, AltaRica is to take benefit from the model-based aspect of Alloy and its expressiveness for the specification of the properties to check. Indeed, Alloy allows to define meta-models easily, which allows for instance to devise domain-specific meta-models. Here, as will be seen, we developed in Alloy a modelling framework called Coy, which can be partly seen as the embedding of the general concepts of AltaRica into Alloy (ignoring concepts we do not need). Furthermore, with Alloy, the specification of the properties we check is expressed in relational first-order logic, with many features adapted to model-based reasoning.

With respect to our previous propositions around using Alloy for MBS&SA, we devise here a richer architectural framework and, more importantly, we formalize a notion of behaviour so as to be able to check properties of the considered system along time.

This section is organized as follows: in §3.1.1, we give a very brief account of Alloy. Then, in §3.1.2, we describe the Coy modelling framework that we implemented in Alloy to model system architectures and their behaviour. We show how Alloy is well adapted to designing domain-specific meta-models and to getting some flexibility in the modelling of time and behaviour. In §3.1.3, we illustrate our approach on a fire detection example that we model in Alloy following the Coy meta-model. In particular, we show how using Alloy allows to express "in one shot" properties ranging over a set of elements selected by navigating in the model structure.

3.1.1 Alloy in a nutshell

Alloy is a formal modelling language that is well adapted to the following (non-exhaustive) list of activities: abstract modelling of a problem or of a system; production of a meta-model (model corresponding to a viewpoint); analysis of a model using well-formedness or formal semantic rules; automatic generation of an instance conforming to a model, possibly according to supplementary constraints; finding interesting instances of a model. Models designed in Alloy can deal with static aspects only, or integrate also dynamic aspects, so as to check behavioural properties.

We now give a brief glance at the main concepts of the language using a simple example. The most important type of declaration is that of a *signature* which introduces a structural concept. It may be seen as a class or entity in modelling parlance. A signature is interpreted as a set of possible instances; and it can also come with fields that may be seen, as a first approximation, as class attributes or associations.

```
sig Data {consumedBy : some System}
sig System {}
sig Criticality {
concernedData : one Data,
  concernedSystem : one System
}
```

[5] See Part A, §2 for short summaries of these publications.

Here, we defined 3 concepts: Data, System and Criticality. Alloy advocates not delving into unnecessary details and only giving information on things we want to understand or analyse. Thus, here, a system is just defined to be a set of “things”, but we do not say anything about the exact nature of its elements.

The keywords `some` or `one` give details on the multiplicity[6] of the relation, as `1..*` and `1` in UML. Here the field declarations mean that: every datum is consumed by at least one (some) system; every criticality concerns exactly one (one) data and one system.

Then, we can add constraints on possible instances of our model. For instance, we would like to state that every system consumes at least one datum. This can be done by writing additional facts (facts are axioms, so as few facts as possible should be stated in order to avoid over-specification):

```
fact {
// every system consumes at least one datum
   all s : System - some consumedBy.s
  // for any system which consumes a given datum, the said
  // datum and system should belong to a same unique criticality
  all d : Data - all s : System | one c : Criticality |
    c.concernedData = d and c.concernedSystem = s
}
```

The `.` operator yields the join of two relations, matching the last column from the first one to the first column of the second one. Thus one may write `d.consumedBy` to get the systems consuming a data “d”, but also “consumedBy.s” to get the data consumed by the system `s`.

The formal foundation of Alloy is relational first order-logic, which is first-order logic extended with relational terms (including the transitive closure of a binary relation). Besides allowing navigation in models, this logic suffices to encode various models of time, *e.g.* to go from a linear to a tree view of time, or to give either an interleaving or a true-concurrency semantics.

Finally, although the language does not preclude unbounded verification in principle, in practice the Alloy Analyzer works only on *finite* models, reducing a given problem to a SAT instance, the analysis of which is delegated to an off-the-shelf SAT solver. Then Alloy may be used to carry out some explorations (the command builds instances that satisfy a given statement) or to check whether a given *assertion* is satisfied by all instances of the model (command). Therefore, as analysis is sound but carried out on finite instances only, the Alloy Analyzer is able to find counter-examples *up to a certain bound* but it cannot prove the validity of an assertion. This is not a problem in our case because:

1) the system architecture we consider is fixed in advance so its number of instances may not vary and
2) only time (*i.e.* the size of the time model) may be unbounded but, in our analyses, we do not aim at proving the absence of errors but rather that a bounded number of events does not lead to a feared situation (which induces that bounded time is sufficient).

3.1.2 The Coy Modelling Framework

We now present the Coy modelling framework, implemented as a meta-model in Alloy, *i.e.* a model where each signature is abstract and only instantiated in a second model corresponding to the system under study. We take inspiration in model-based safety assessment but our formalization is not specific to this sole family of properties.

As will be seen hereafter, Coy models essentially represent hierarchical structures of transition systems communicating instantaneously through data ports.

The overall structure of the framework is presented graphically in the next figure. Extension links are figured using black dashed arrows. As the meta-model contains *n*-ary relations with *n* > 2, the figure shows these after projection on parts of their domain (this is indicated using square brackets, as in `conns[Port]` for instance). Furthermore, the meta-model contains a “Time” signature: its purpose is that every signature field with `Time` as its last column can be conceptually seen as a mutable field, *i.e.* its value may change (discretely) over time. Notice that the meta-model in the following figure is projected over `Time`, hence it is not shown in the diagram.

6 Other possible multiplicities are: lone which means at most one (0..1); and set which means any number (0..*).

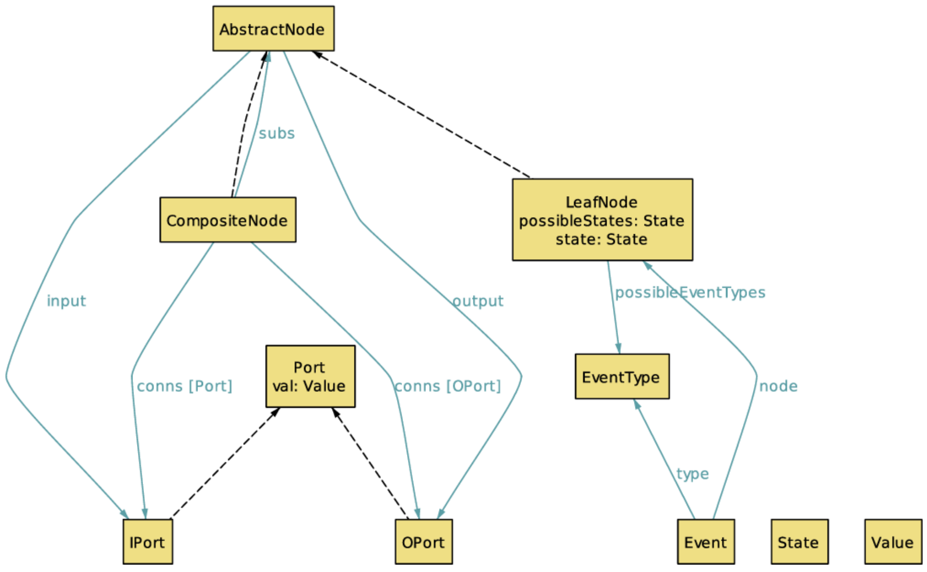


Figure 4: Graphical depiction of the Coy meta-model (projected over Time)

3.1.2.1 Composite Structure

Let us now delve into more details in the meta-model. In what follows, for the sake of readability, we do not show all Alloy facts enforcing the well-formedness of instance models or just classical properties. The basic architectural element is a *node*. Nodes are arranged hierarchically as a tree, so we use the classical *Composite* design pattern and devise a notion of `AbstractNode` which is inherited by signatures `CompositeNode` and `LeafNode`, the former pointing back to abstract nodes.

Every node comes with a set `IPort` of input ports and a set `OPort` of output ports. These sets are disjoint and every port belongs to a single node. Every port carries a value at every instant (possible values may differ for distinct ports).

Connections (between ports) are constrained so that they cannot cross a parent node boundary or many levels of composition. In other words, nodes are arranged as trees and connections can only happen between siblings or between a parent and a child. Furthermore, connected ports always carry the same value.

```
abstract sig Port {
  // a port carries one value at every instant
   Value one -> Time,
}
abstract sig IPort, OPort extends Port {}
abstract sig AbstractNode { // input and output ports
  input : set IPort,
  output : set Oport,
}
abstract sig CompositeNode extends AbstractNode {
   // a composite node contains at least one sub-node
  subs : some AbstractNode,
  // port connections with siblings and between sub-nodes
  //and this node
   conns : subs.@output -> subs.@input
         + input -> subs.@input
         + subs.@output -> output,
} {      // connected ports always carry the same value
  all t : Time, po, pi : Port - po->pi in conns
                        implies po.val.t = pi.val.t
  // + other structural properties
   ...
}
abstract sig LeafNode extends AbstractNode {...}
```

3.1.2.2 Behaviour

As Coy is mainly aimed at describing systems where atomic nodes are endowed with behaviour, we now introduce a notion of state (for leaf nodes) and of events that may happen. One approach to deal with such models could be to rely on classical model-checkers, such as Spin or NuSMV, the modelling languages of which are well-suited for describing transition systems. While this is of course a possibility, our aim with using Alloy is:

- to be able to easily adapt the Coy meta-model depending on the domain of study (*e.g.* to add a notion of connectors as in many architecture-description languages);

- as explained earlier, to change the model of time if need be (*e.g.* to go from a linear to a tree view of time, or to give either an interleaving or a true-concurrency semantics);
- and above all, to allow the use of logic for specification, which brings two interesting aspects:
  - it provides a single language to specify both models and expected properties;
  - the logic allows for the expression of rather abstract properties (*e.g.* relying on under-specification) or to navigate through elements of a model to specify a given property in only one formula.

Thus, every leaf node is in one given state at any time (the set of possible states may vary for two different nodes). Besides, such nodes may undergo events. An event is an instance of an event type that happens at a certain instant and concerns a given node: this distinction between events and event types then allows to consider events of a certain type only, for instance to characterize their effects.

Notice also we impose the end-user to give, for any leaf node, the set of its possible states and the set of event types that concern it: this is a bit redundant from the theoretical point of view but it provides a sort of additional safety check akin to a poor man's typing that we deem important from a methodological point of view.

```
abstract sig LeafNode extends AbstractNode {
 possibleStates : some State,
 state : possibleStates one -> Time,
 possibleEventTypes : set EventType,
}
abstract sig EventType {}
abstract sig Event {// event occurrence
 instant : one Time,
 node : one LeafNode,
 type : one EventType
} { type in node.possibleEventTypes }
```

Finally, as Alloy does not feature a native notion of time, we encode it by characterizing finite traces of instants. The fact accounting for this says how states change depending on events, at every instant.

```
fact traces {
//if a node state changed, there was an event concerning this node
 all t : Time, t' : t.next, n : LeafNode |
        n.state.t != n.state.t' implies some e : Event |
                e.instant = t and e.node = n
}
```

In practice, in the context of safety and security engineering, failures and threat scenarios are modelled as event types, as we will see in the next section. Their effect on a node must be described as an Alloy fact.

3.1.3 Fire detection example

In this section, we provide an illustration of Coy with a fire detection system in a facility such as, for example, an airport or a port.

3.1.3.1 Presentation of the system

The system consists of the following components: a *smoke detector* and a *heat detector*, which are part of the automatic *fire alarm system*; a manual *fire alarm pull station*; the *local firemen*, inside the facility; and the *city firemen*, in the nearest city.

The automatic fire alarm system, which is activated by either of the two detectors, directly calls the city firemen. The manual pull station, triggered by a human present on site, calls both the local and the city firemen.

We also represent two possible failures for each of the components: (1) the loss of a component: once a component is lost, it does not send any information, (2) an erroneous failure of a component: after this kind of failure, a component sends a corrupted data (in the case of a fire detector, for instance, it can be a false alarm or a false negative). Lastly, we represent three security threats scenarios (called simply threats in the reminder): (1) intentional wrong activation of the pull station, (2) the deactivation of the smoke detector and (3) of the heat detector.

Notice that the loss of a component and the deactivation of the smoke detector (or of the heat detector) have the same effect on a component (the availability is not ensured) although they do not have the same nature (the former is a failure, the latter is a security threat). The same applies to an erroneous failure and the intentional wrong activation of the pull station, which both affect the integrity of the information. Nevertheless, it is important to distinguish between these failure and threat events in order to allow a pure safety analysis, a pure security analysis, and a combined analysis.

3.1.3.2 Coy model

The Coy model of this system imports the Coy meta-model, declares instances of signatures and relates them. Components of the system are modelled as Coy nodes. The following Alloy code illustrates a particular instance of the fire detection model at a given instant. As can be seen, at this instant, that all nodes are in the state `OK` and that all ports yield a correct data (modelled by `OKVal`). The occurrence of an event of type `failLoss` (see the declaration of event types below) can be observed on the node `pullStation`.

Regarding the possible failures and threats mentioned above, we use the following event types, node states and possible values for the node ports.

```
one sig failLoss, failErr, threatBlock, threatPull
        extends EventType {}
one sig OK, Lost, Err extends State {}
one sig OKVal, LostVal, ErrVal extends Value {}
```

Then, we can declare the components and ports, as instances of the corresponding Coy concepts. For instance, here is the declaration of the fire pull station.

```
one sig pullStation extends LeafNode {} {
 input = none and output = oPullStation
 possibleStates = OK +Lost +Err
 and possibleEventTypes = failLoss +threatPull
}
```

The model also comprises axioms stating what happens to nodes depending on observed events. An interesting point here is that this description is declarative and does not depend on the effective nodes and ports. Concerning an event of type `failLoss`:

- the event can only occur on a node which is not in the state `Lost`,
- after the occurrence of the event, the node moves to the state `Lost`.

Here, we chose to model events of type `threatBlock` in the same way (the node also moves to the state `Lost`). So, they have the same effect (but they do not occur on the same components). In further analysis, if we want to distinguish between the effects of both kinds of events, we just have to use a specific node state and a specific port value corresponding to the occurrence of `threatBlock`.

The behaviour of events of type `failErr` and `threatPull` are specified in a similar way.

```
fact behaviour {
 all e : Event | e.type in failLoss +threatBlock
        implies e.node.state.(e.instant) =Lost
               and e.node.state.(e.instant.next) = Lost
 all e : Event | e.type in failErr +threatPull
        implies e.node.state.(e.instant) = OK
               and e.node.state.(e.instant.next) = Err
}
```

The propagation of values is also described by an Alloy fact. For example, here is the description of the value propagation for leaf nodes with one input:

```
// leaf nodes w/ 1 input
all n : LeafNode, t : Time | {
 one n.output   // tautology for this specific model,
                      // but useful if we extend it
 one n.input
} implies {
 n.state.t = OK implies n.output.val.t = n.input.val.t
 n.state.t = Err implies n.output.val.t = ErrVal
 n.state.t = Lost implies n.output.val.t = LostVal
}
```

3.1.3.3 Properties verification

Now we can express the safety and security properties that we want to check as Alloy *assertions*. We have mainly expressed properties related to the consequence of some failures/threats or to the robustness of the system to a given number of failures/threats. For instance, the following assertion states that whenever the smoke detector is lost (and all other nodes are OK) then the firemen can still act.

```
assert smokeDetectorLoss {
 all t : Time | {
 all n : LeafNode - smokeDetector | n.state.t = OK
        smokeDetector.state.t = Lost
 }implies (localFiremen +CityFiremen).output.val.t = OKVal
```

```
    }
```

The following assertion expresses that whenever the pull station is attacked, (and all other nodes are OK) then at least one firemen department is able to act.

```
    assert pullStationThreatPull{
     all t : Time | {
             all n : LeafNode - heatDetector | n.state.t = OK
             heatDetector.state.t = Err
     } implies OKVal in (localFiremen +CityFiremen).output.val.t
    }
```

Notice that the first-order quantifiers and the object-oriented syntax allow navigating easily through the model and are convenient to state safety properties.

The following assertion expresses that whenever the smoke detector is erroneous and the pull station is attacked threatPull, then both local and city firemen are unable to act.

```
    assert smokeDetectorFailErrPullStationThreatPull {
     all t : Time | {
             all n : LeafNode - (smokeDetector +pullStation)
                     | n.state.t = OK
                     smokeDetector.state.t = Err
                     pullStation.state.t = Err
     }implies OKVal not in (localFiremen +CityFiremen).output.val.t
    }
```

The following assertions express the robustness of some parts of the system to possible failures/threats. We took benefit from the possibility to reason about a set cardinality in Alloy. Here, we count the number of events (corresponding to failures/threats) that occurred before the system enters a bad situation. For instance, the following assertion expresses that in order to make both local and city firemen unable to act properly, either the threat threatPull has occurred, or there has been at least two distinct failures/threats. Remark that it would have been also possible to reason independently about the number of failures and about the number of threats.

```
    assert noSingleFailureThreatLeadsToFiremenNotOK {
     all t : Time |
     OKVal not in (localFiremen +CityFiremen).output.val.t
     implies some e : Event |
             e.type = threatPull and lt[e.instant, t]
             or let events = { e : Event | lt[e.instant, t] } |
             #events ≥ 2
    }
```

In order to check assertions, Alloy Analyzer searches for counter-examples up to a certain bound (*i.e.* the counter-examples are such that their signatures have a cardinality less than the bound). The bound can be given by the user or chosen by the tool. In general, this bounded verification is thus incomplete: the tool may not find counter-examples whereas there are some. But in our case, the cardinality of all the signatures (nodes, ports, etc.) is fixed by the model itself. Therefore, the verification performed by the Alloy Analyzer is complete.

The aforementioned four assertions have been validated by the Alloy Analyzer (i.e. it did not find any counter-example).

The following assertion expresses that in order to make both local and city firemen unable to act, there has to be at least three failures/threats in the architecture.

```
    assert noDoubleFailureThreatLeadsToFiremenNotOK {
     all t : Time |
     OKVal not in (localFiremen +CityFiremen).output.val.t
     implies let events = { e : Event | lt[e.instant, t] } |
              #events ≥ 3
    }
```

This last assertion is not satisfied by the model. Alloy Analyzer exhibits a counter-example where the pull station and the city firemen are lost after two events. The following figures show respectively the first time step and the second time step of this counter-example. In these figures, leaf nodes are beige rectangles, output ports are red trapeziums, input ports are green trapeziums and connections between ports are blue arrows.

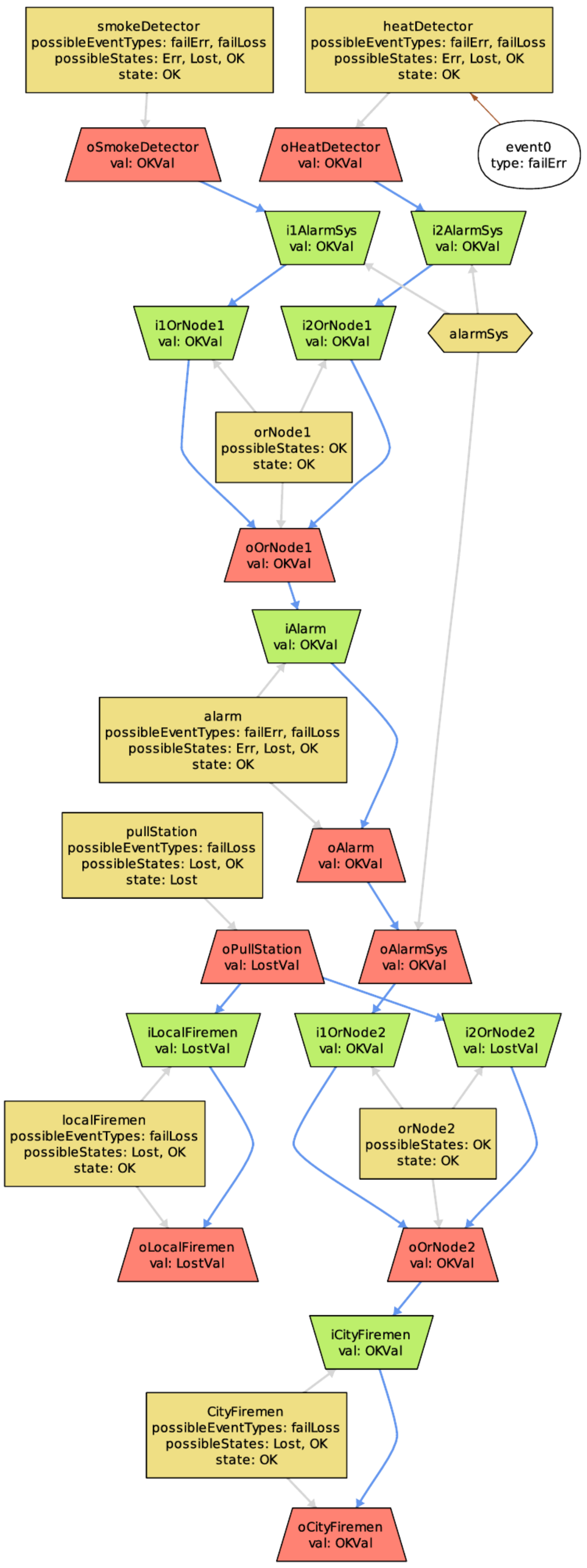


Figure 5: Second time step of the counter example (failErr on the heat detector)

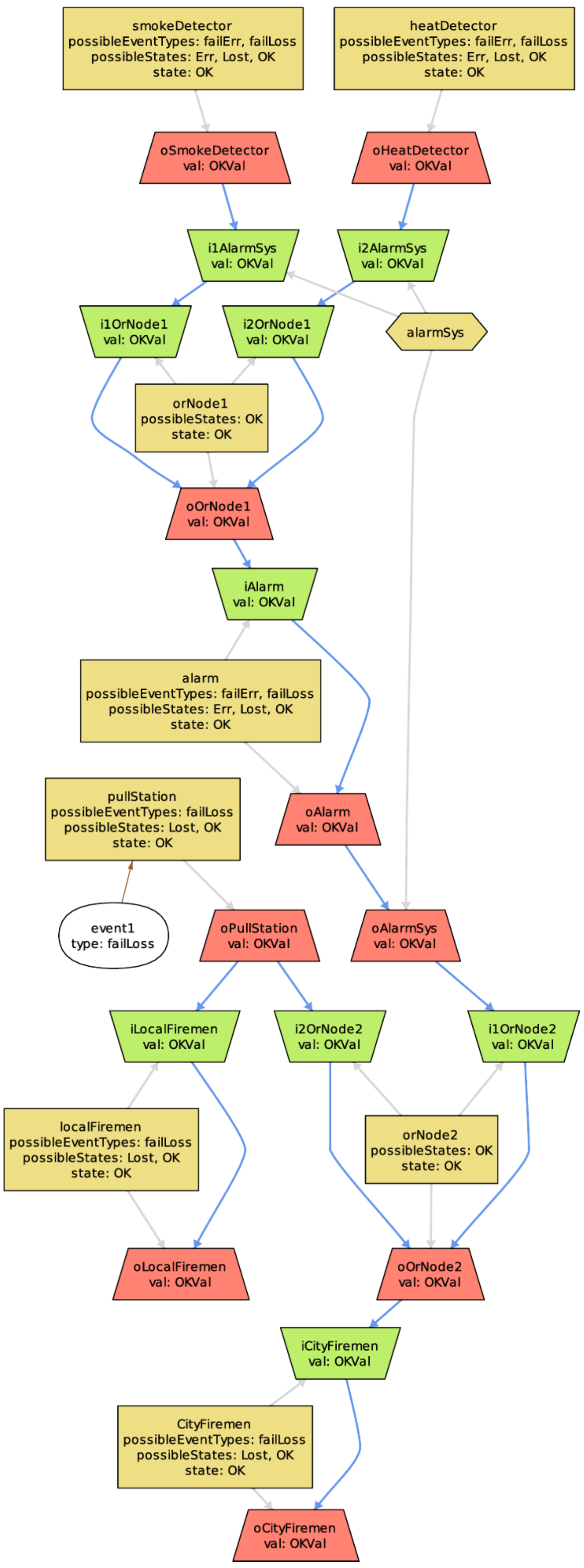


Figure 6: First time step of the counter-example (failLoss on the pull station)

3.2 Integrated approach

We have proposed and prototyped an approach which consists in decoupling the system architecture model from safety & security models (cf. Figure 7). In this approach, every engineer, whether architect, or security / safety engineer, can focus solely on his concerns, with dedicated tools and terminology.

As of now, we chose to use two separate models: one for the safety concern and the second for the security concern. The main motivation for this separation is that safety and security domains are quite different in terms of practices, concepts used and wording.

As the safety and security models rely on the system architecture model, we extract the required information (e.g. function interactions, ports and their links, data…) from the architecture model and we set up initial safety and security models in Safety Architect. Starting from this, safety and security engineers complete their model by adding safety and security dysfunctional behaviour. The safety and security models contain two kinds of information:

- the safety and/or security dysfunctional behaviours; a safety dysfunctional behaviour represents how errors are propagated in the system architecture, and a security dysfunctional behaviour represents how the effects of security threats are propagated in the system architecture;
- the safety and/or security properties / requirements that the system architecture must satisfy, e.g. the integrity of the output data shall be preserved even under specific attacks.

These two models are then combined to produce a formal Alloy model containing all the necessary inputs for the analysis. The Kodkod tool formally validates the safety and security properties. If a property is violated, Kodkod will show a readable counter-example. Thus, the engineers can identify the best way to correct the architecture.

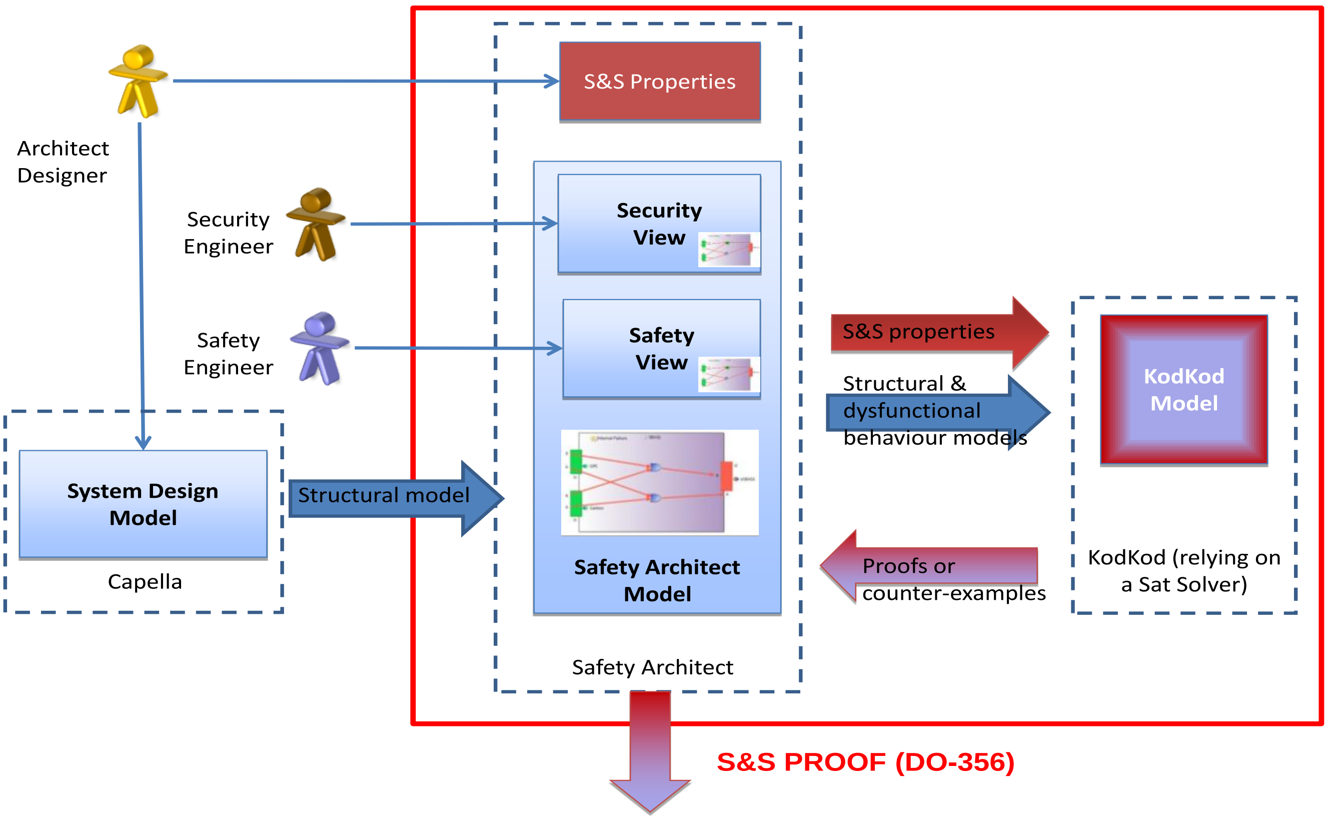


Figure 7: Integrated approach

The main principles of this approach are illustrated in Figure 7. The first model transformation yields an initial Safety Architect model from the System Design Model. This transformation is trivial as it only reflects the structural part of the architecture. Capella functions are mapped to Safety Architect blocks. Ports give input and output ports, while data links yield data links.

Then, safety engineers and security engineers can work within Safety Architect, either using separated views or a merged view, to describe the way failures and security threats propagate inside the architecture: this activity is called dysfunctional modelling. Then dysfunctional analysis techniques already available in Safety Architect can be applied, such as the automatic generation of fault trees or attack trees.

The last part of the approach consists in taking benefits from the Alloy formal specification and verification method, and its relying Java Kodkod API, to enhance the analyses we can perform. The idea is to build a

Kodkod model that represents the architecture of the system and the rules that describe the way failures and threats propagate inside each block, and along the architecture. Then we can use all the expressive power of the Alloy logic and the verification capability of Alloy Analyzer/Kodkod to check various kinds of properties. In the approach we propose, the building of the Kodkod model and the verification of formal properties are proto-typed and called from Safety Architect.

3.2.1 Alloy/Kodkod model

Although our prototype uses Kodkod, it is more convenient to explain the main ideas behind the Kodkod model in Alloy terms, because Alloy is a language, whereas Kodkod is a Java API.

The main concepts of Safety Architect are declared as Alloy signatures and fields[7]. For example, Figure 8 shows an excerpt of the graphical view of the Alloy model corresponding to the structural part of the Safety Architect model, in which blocks, ports, input and output ports are defined as signatures, whereas values (i.e. assigning a status to a port) ins, outs (i.e. associating each block with its input and output ports) and binds (i.e. representing bindings between ports) are defined as fields.

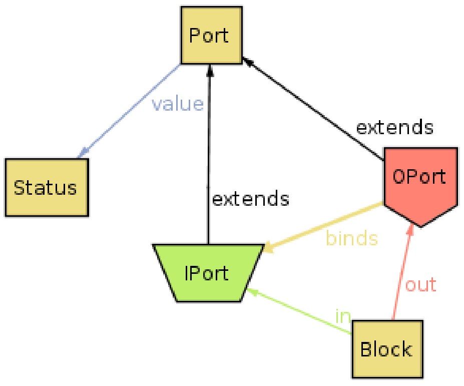


Figure 8: Alloy/Kodkod model

We also define signatures and fields to represent all the concepts of Safety Architect, such as barriers, internal failures, and propagation trees, which express the way threats and failures propagate inside each block, including the effect of barriers and internal failures.

Now, in a particular instance of an Alloy/Kodkod model, each port of each bock in the architecture is associated with a particular status (either *OK*, or one of the failure modes that are declared). The statuses of ports have to respect the following constraints:

- if two ports are bound, then they have the same status,
- the statuses of ports respect the propagation trees inside each block.

As an illustration, the former constraint is expressed in Alloy as follows:

```
all op : Oport | all ip : op.binds | op.value = ip.value
```

In this formula, the term *op.binds* represents all the input ports that are associated with *op* through the field *binds*.

3.2.2 Formal proofs

The main goal of building a Kodkod model from a Safety Architect model is to be able to apply formal verification. We call property, the safety and security properties that the architect is interested in verifying on a given system model. In this section, we present two classes of properties that we can check easily with our approach, in the sense that their verification does not require a high level of Alloy or Kodkod expertise. Other kinds of properties can be verified, but they need to be specified directly in Java, using the Kodkod API, and this requires specific expertise.

7 Please refer to §3.1.1 for a definition of signature and field.

3.2.2.1 Structural properties

This first class of properties deals with the structure of the model (i.e. the blocks, their ports, the bindings between ports) but not with the safety- or security-related aspects (i.e. statuses of ports, feared events…). More precisely, it reflects a certain form of correction in the binding of ports:

```
If an output port op is bound to an input port ip, then the possible statuses of op and ip (de-
fined by all the propagation trees that lead to op and all the propagation trees that take ip as
input) are the same.
```

This property is easily defined in Alloy/Kodkod as a formula. Since this formula relies on the signatures and fields related to the propagation trees, we do not express it in this document. It is defined in Kodkod in our prototype. Note that the non-satisfaction of this constraint is not an error: it raises a warning that points out a possible conceptual problem in the model.

Other kinds of structural properties could also be expressed and checked in Java using the Kodkod API. But they are not predefined in the prototype and would then require a high level of expertise in Kodkod.

3.2.2.2 Properties related to feared events

The second class of properties that we handle is related to feared events.

First, a feared event needs to be defined as a Boolean combination of statuses of ports, e.g.:

```
The output port op1 has the status lost and the output port op2 has the status erroneous.
```

This concept is defined in Kodkod in our prototype, which allows the engineer to express a number of properties that he wants to check over the architecture related to feared events, provided he has an expertise in Kodkod.

The specific class of properties that we defined in our prototype (in order to ease its verification by an engineer) concerns the minimum number of failures that is necessary to reach a feared event: the method *atLeastNFailures(FearedEvent fe, int n)* is defined as a formula expressing the following statement in Alloy/Kodkod logic :

```
If a feared event occurs, then the set of failures that occurred is greater than n.
```

We can also express in Kodkod properties about the consequences of a threat or a failure, or about the impact of a given (set of) threat(s) or failure(s) on a feared event, etc., but this would require an expertise in Kodkod.

Kodkod is able to formally check these properties and it answers:

- either “correct”, if the property is true for every possible instance of the architecture, i.e., exploring for each port every possible status that is compliant with the aforementioned constraints of the Kodkod model,
- or “not correct”, in which case, it shows an instance that violates the property.

3.3 Lessons learnt

In this chapter, we have experimented different formal techniques relying on the Alloy method to support safety and security co-engineering.

First, we have proposed a standalone approach, Coy, in which the users (i.e. safety and security engineers) are expected to model the system architecture and the properties to check using the Alloy language. This work requires specific expertise.

Then, we have proposed an approach that relies, from the user point of view, on existing industrial tools: *Capella* for architecture design and *Safety Architect* for safety and security engineering. The formal verification capabilities are directly available from (an extension of) Safety Architect. Some of the verification queries are predefined and easy to call from Safety Architect. For other formal verification activities, which use the underlying Kodkod model, the user must master the Kodkod language.

These two approaches allow for the generation of a proof that certain properties related to feared events are fulfilled, and allow for the generation and study of scenarios that violate a property, if there are some. In the first approach, Coy allows specifying rich architecture models and offers a high expressiveness in terms of properties to verify. This is particularly true if the temporal evolution of the system architecture or the temporal ordering of failure/threat occurrences matters. In the second approach, the expressiveness is limited, e.g. the architecture model is purely static, the user cannot add new concepts to the language, etc. But in this second approach, an expertise in Alloy is not needed, at least to check the properties that are pre-defined in the approach.

4 An analysis of a third-party tool: TTool/AVATAR

Following the state of the art work presented in Part A of this deliverable, TTool/AVATAR appeared to us as one of the most interesting tools readily available. We have therefore decided to perform an in-depth analysis of the tool.

The Automated Verification of reAl Time softwARe tooled-up profile (AVATAR, 2015) is an extension of SysML (System Modelling Language), provided by Telecom Paris-Tech, enabling the formal modelling and verification of safety and security properties.

TTool, the toolkit implementing the AVATAR profile, transforms the AVATAR model into the appropriate timed automata or pi-calculus model, launches UPPAAL or ProVerif in the background to verify respectively safety and security properties, and provides the designer with the results, as a textual display and / or as back-annotations on the AVATAR model. The simplicity of the profile and the tool's ergonomics enable system / software designers with a generic UML/SysML background to formally model and verify safety and security properties, where formal method experts are classically needed.

Even though it is possible to develop an application completely using AVATAR, the goal of AVATAR is essentially to model the control logic of an application in order to verify its safety and security properties. The use of AVATAR is thus compatible with the existence of a legacy system / software engineering workbench.

With respect to security, AVATAR supports the modelling of confidentiality and authenticity properties. Security properties are modelled as SysML pragmas, thus inducing no significant formal modelling overhead; in addition, formal verification is straightforward, allowing formal proofs to be produced at little costs. The scope of application is the definition of new secure communication protocols, or the securing of existing communication protocols. It is to be noted that the relevance of AVATAR may be limited for companies using standard communication protocols.

With respect to safety, AVATAR supports the modelling of any safety property, through the use of SysML observer blocs and the reachability and / or liveness of some of the observer states. Temporal operators allow for the modelling of complex safety properties for real-time systems. The scope of application is therefore very wide, and should be usable by any company designing safety-critical systems. However, the current modelling approach based on SysML represents a significant work overhead; in the long run, safety properties should be modelled in AVATAR using the TEPE (TEmporal Property Expression) language, bringing down the formal modelling overhead to an acceptable level.

A complete evaluation of AVATAR is provided below.

4.1 Introduction to the AVATAR UML / SysML profile

The AVATAR profile (Apvrille, 2015) stands for Automated Verification of reAl Time softwARe. It targets the modelling and formal verification[8] of real-time embedded systems.

To support the use of UML and SysML models in a verification-centric method, the AVATAR profile rests upon:

- a toolkit called TTool for the edition and the simulation[9] of extended UML and SysML diagrams;
- a formal property expression language called TEPE, for TEmporal Property Expression language;
- the third-party UPPAAL and ProVerif tools dedicated to the formal validation of safety and security properties, as expressed in the AVATAR diagrams.

TTool (pronounce "tea-tool"), whose first version was published in 2003, currently supports several UML profiles, including:

- AVATAR[10]: UML 2.x and SysML-based profile for the modelling and formal verification of real-time embedded systems;
- DIPLODOCUS: UML profile dedicated to the partitioning of Systems-on-Chip;
- Network Calculus: profile dedicated to the dimensioning of critical systems;
- CTTool: profile dedicated to the modelling and verification of component-based and distributed systems;

[8] We insist here on the formal verification, rather than on the modelling, because modelling may be limited to the minimal effort necessary to support formal verification.

[9] Simulation capabilities are not detailed in this document. For the end-user interested by this feature, it is to be noted that method calls are not executed during the simulation.

[10] AVATAR replaces TURTLE (Timed UML and RT-LOTOS Environment). TURTLE was a UML 1.5 profile targeting the modelling and formal verification of real-time embedded systems, and is obsolete since end 2010.

- SysML-sec, a profile created with a sub-part of AVATAR and a sub-part of DIPLODOCUS, with a new methodology.

TTool and most of its profiles are provided by Télécom Paris Tech[11]. They have given way to multiple publications, all referenced and availble on the TTool web site (Apvrille, 2015). They are used for teaching at EURECOM (“Ecole d’ingénieur et centre the recherche en télécommunications”). TTool’s external references include Thales Alenia Space[12], UDcast, Texas Instruments[13], Freescale, the EVITA project, the SACRA project, ISAE, LAAS-CNRS, Tesa, INRIA, and DOCEA Power. The latest version of the TTool package (including the different profiles) is v0.97, published in January 28th, 2015. This shows that the tool has regularly evolved and has been maintained for close to 15 years. During our tests, we started with version 0.97, then switched to the beta version of 0.98, which allowed us to benefit from the latest features and some specific patches.

The TEmporal Property Expression (TEPE) language (Knorreck, et al., 2010) is a graphical and formal language based on SysML parametric diagrams provided by Télécom Paris Technical and seamlessly integrated to AVATAR. TEPE allows for the expression of physical time and unordered signal reception. In TEPE, various design elements, such as SysML blocks, attributes, and signals, can be combined together with logical (e.g., sequence of signals) and temporal operators (e.g., a time interval for receiving a signal) to build up complex but graphical properties. The strength of the AVATAR-TEPE combination is that requirement capture, analysis, design, property description and verification tasks can seamlessly be accomplished in the same TTool environment. The designer is merely required to have minor UML skills and does not need to be familiarized with formal languages like CTL or UPPAAL.

4.1.1 Third-party companion tools

UPPAAL (UP4ALL) is an integrated tool environment for modelling, validation and verification of real-time systems modelled as networks of timed automata. UPPAAL consists of three main parts: a description language, a simulator and a model-checker. The description language is a non-deterministic guarded command language with data types, e.g. bounded integers, arrays. It serves as a modelling or design language to describe system behaviour as networks of automata extended with clock and data variables. The simulator is a validation tool which enables examination of possible dynamic executions of a system during early design (or modelling) stages and thus provides an inexpensive mean of fault detection prior to verification by the model-checker, which covers the exhaustive dynamic behaviour of the system. The model-checker can check invariant and reachability properties by exploring the state-space of a system, i.e. reachability analysis in terms of symbolic states represented by constraints. The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden, and the Department of Computer Science at Aalborg University in Denmark. UPPAAL is commercialised through a spin-off called UP4ALL International AB.

ProVerif (Blanchet) is a toolkit that relies on Horn clauses resolution for the automated analysis of security properties over cryptographic protocols. ProVerif takes as input a set of Horn Clauses, or a specification in pi-calculus (process algebra) and a set of queries. ProVerif outputs whether each query is satisfied or not. In the latter case, ProVerif tries to identify a trace explaining how it came to the conclusion that a query is not satisfied. In ProVerif, a specification takes the form of a system represented as spi-calculus processes, and properties are represented as queries. Queries can be used to express confidentiality[14] and authenticity[15] requirements. ProVerif also makes it possible to study the reachability of events, based on queries; therefore, ProVerif may also be used for proving safety properties on the system augmented with the attacker. ProVerif integrates its own attacker model, which is a process implementing a Dolev-Yao approach (Dolev, et al., 1983). ProVerif is provided by INRIA.

4.1.2 Licences

TTool is a software computer program distributed under two licenses:

- BSD+ license for some source files and icons, distributed by SUN Microsystems;

[11] Ludovic Apvrille is the main developer of AVATAR, with some contributions by a few students; one LIP6 permanent and one LIP6 student are also currently contributing on the deployment features. On SysML-sec, there are currently multiple PhD students working, and some code produced by Freescale has been integrated, the lot being validated by many significant test cases. AVATAR should directly benefit from the work on SysML-sec. In the past, there have been contributions to AVATAR by Freescale, PhD students and students.

[12] According to Isabelle Buret (TAS), the experience with TURTLE, the ancestor of AVATAR, was non-conclusive. The approach was deemed too singular or “avant-garde” for the target, i.e. mission-critical embedded software for satellites.

[13] Texas Instruments funded the development of DIPLODOCUS for 5 years, without directly contributing to the code, but showing interest in the results.

[14] Confidentiality queries directly express which data must not be accessible to the attacker, e.g., that a private key shall not be accessible to an attacker: *query attacker:myKey*.

[15] Authenticity of messages relies on ProVerif events and correspondence assertions. Whenever a message m, sent by a process A to a process B, must be authenticated, one event is included in each process: one is included in A before the sending of m (e.g., eventSendM), and one after the receiving of m (e.g., eventReceiveM). Since the attacker is not allowed to execute events, it suffices to prove that to each receiving event of m corresponds exactly one sending of m. Thus, an injective query is used to model authenticity: *query evinj:eventReceiveM(x) ==> evinj:eventSendM(x)*.

- CeCILL license for all other parts.

UPPAAL is available as a commercial version for industry and national research agencies, or as a free academic version, for academic users.

ProVerif, is distributer either as a source package under the GNU General Public License, or as a binary package for Windows, under the BSD license.

4.1.3 Tool installation and configuration assessment

Complete instructions for the installation and configuration of TTool / AVATAR are given on the web site (Apvrille, 2015). Supported platforms include Windows and Unix (incl. MacOS).

We performed the installation on Windows XP, and later on Windows 7.

The installation was straightforward, including for third-party companion tools, i.e. UPPAAL and ProVerif.

4.1.4 User manual

The AVATAR profile for TTool has no user manual as such. A lot of guidance can be found on the web site (Apvrille, 2015), including material for EURECOM students (Apvrille, 2014) (Apvrille, et al.). However, the material is incomplete, and often obsolete.

To proceed, we contacted Telecom Paris Technical directly. Excellent tool support was found through direct contact with Prof. Ludovic Apvrille, and a PhD student, Florian Lugou. They provided advice, examples and even patches within hours.

4.2 Methodology assessment

The AVATAR profile for TTool comes with its own methodology. It supports six[16] methodological phases (cf. Figure 10):

- **requirements capture**: requirements and properties are structured using the AVATAR requirement diagrams, as per SysML; at this stage, requirements are just defined with a specific label – see the *property modelling* phase below for the formalisation of the requirements; during this phase, system and environmental assumptions may be captured using AVATAR Modelling Assumptions Diagrams (MADs).
- **system analysis**: a system may be analysed using usual UML diagrams, i.e. use case, context, activity and sequence diagrams; the system may also be analysed using attack trees, but since the AVATAR attack trees involve Blocks, they need to be developed iteratively with the system design;
- **system design**: the system is designed in terms of communicating SysML blocks described in an AVATAR block diagram, and in terms of behaviours described with AVATAR state machines[17];
- **property modelling**: the formal semantics of properties is defined using AVATAR property diagrams; modelling is performed using the TEmporal Property Expression (TEPE) language within Parametric Diagrams (PDs); since TEPE PDs involve elements defined in the system design (e.g, a given integer attribute of a block), TEPE PDs must be defined iteratively with the system design;
- **formal verification**: this phase can be conducted over the system design, and for each test case defined in the requirement diagrams;
- **code generation**: can finally be used to generate a fully executable code; the latter can be compiled and executed on the SoCLib prototyping platform directly from TTool.

Property modelling, using the TEPE language, comes with its own methodology. The main lines are given below. For more details concerning the TEPE language concepts, please refer to Annex B, or to (Knorreck, et al., 2010).

[16] The (obsolete) TURTLE profile supported four methodological phases: requirements capture, system analysis, system design, and deployment.

[17] A state-machine diagram is automatically created for each new block.

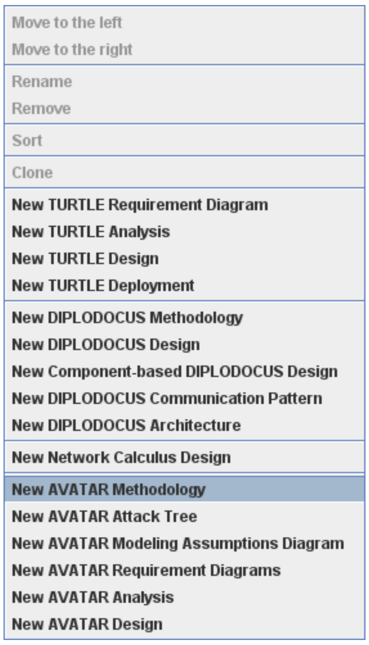

Move to the left
Move to the right
Rename
Remove
Sort
Clone
New TURTLE Requirement Diagram
New TURTLE Analysis
New TURTLE Design
New TURTLE Deployment
New DIPLODOCUS Methodology
New DIPLODOCUS Design
New Component-based DIPLODOCUS Design
New DIPLODOCUS Communication Pattern
New DIPLODOCUS Architecture
New Network Calculus Design
New AVATAR Methodology
New AVATAR Attack Tree
New AVATAR Modeling Assumptions Diagram
New AVATAR Requirement Diagrams
New AVATAR Analysis
New AVATAR Design

Figure 9: Creating a diagram for the methodology

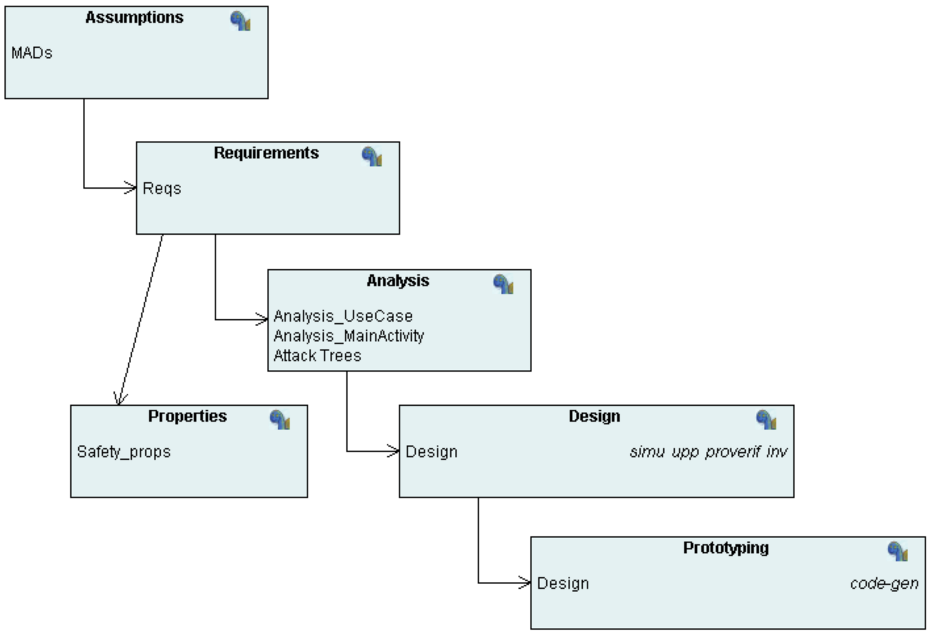


Figure 10: The six phases of the AVATAR methodology

A TEPE parametric diagram[18] is supposed to be constructed in the following way:

- First, blocks are represented with their particular attributes and signals subject to verification. These entities should normally have been identified during the design phase.
- Values derived from original attributes and signals are introduced (using equation and alias operators).
- The reasoning about the sequential and temporal behaviour of the system is expressed in terms of logical and temporal operators connected to signals and properties. These logical and temporal operators can be cascaded.
- Several properties may be merged using logical property operators (conjunction, disjunction, property definition operators).
- Finally the formal property is labelled to link it to an informal SysML requirements diagram and to determine whether (non-) liveness or (non-) reachability should be verified on that property.

To avoid overloaded diagrams, the constituting properties of a requirement can be spread over several diagrams.

Assessment take-away

The AVATAR profile is provided with a methodology and its integration inside the tool as a specific diagram with a traceability feature to other diagrams has been much appreciated.

However, the methodology would require some fine tuning to deal with dependencies between steps, and would gain in being more detailed.

4.3 Tool assessment

The tool assessment is performed using a simple running example and follows the recommended methodology (cf. §4.2).

The running example is a case of a microwave oven that is used to heat meals for a defined duration on the press of a button, or via a remote control command. Whenever the door is opened, the magnetron must be turned off (safety constraint), and to avoid an overload of the magnetron, it should not be operated more than 10 units of time at full power (safety constraint). The remote control must be secured, that is, a remote control must

[18] In TTool, the AVATAR property diagrams are organised under the umbrella of the requirements diagrams.

be attached to only one specific oven (authenticity constraint), and messages sent from the remote control must not be disclosed[19] (confidentiality constraint).

Using the tool, a diagram can be created (cf. Figure 9) with the phases of the methodology (cf. Figure 10). An interesting feature is that each phase of the methodology can list the diagrams that are related to it. A minor issue is that the steps of the methodology diagram are not identical to the steps described in the methodology documentation (Apvrille, 2015), even if a mapping may be extrapolated.

4.3.1 Elicitation of assumptions

AVATAR allows for the capture of assumptions. Assumptions are characterised by the following tags:

- Type: it is possible to choose between two stereotypes, namely <<System Assumption>> and <<Environment Assumption>>;
- Name: identifier (spaces are not allowed);
- Text: it allows for the description of the assumption in an informal way;
- Durability: this tag may take for value Undefined, Permanent or Temporary;
- Source: this tag may take for value Undefined, End-user, Stakeholder, or Model creator;
- Status: this tag may take for value Undefined, Applied or Alleviated;
- Scope: this tag may take for value Undefined, Language, Tool, Modelling activity or Verification.

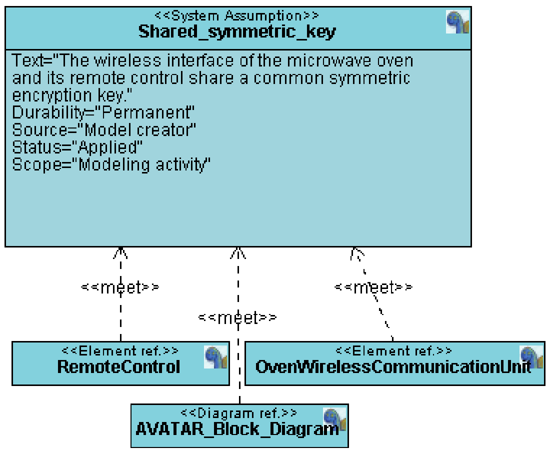


Figure 11: Example of AVATAR Modelling Assumption Diagram

In addition to the possibility of creating assumptions, the AVATAR Modelling Assumptions Diagram allows for:

- decomposing assumptions;
- versioning assumptions;
- adding references to diagrams and / or model elements that meet those assumptions.

Figure 11 shows an example of an AVATAR Modelling Assumption Diagram with the assumption that "*The wireless interface of the microwave oven and its remote control share a common symmetric encryption key*". The assumption can be traced for its implementation to the block diagram in general, and two of its blocks in particular (cf. §4.3.4).

Assessment take-away

The AVATAR capability of explicitly capturing assumptions and tracing them to design elements is deemed very important.

However, many properties in AVATAR are captured as pragmas, and traceability is not supported to UML comments. This needs to be enhanced.

4.3.2 Requirements capture

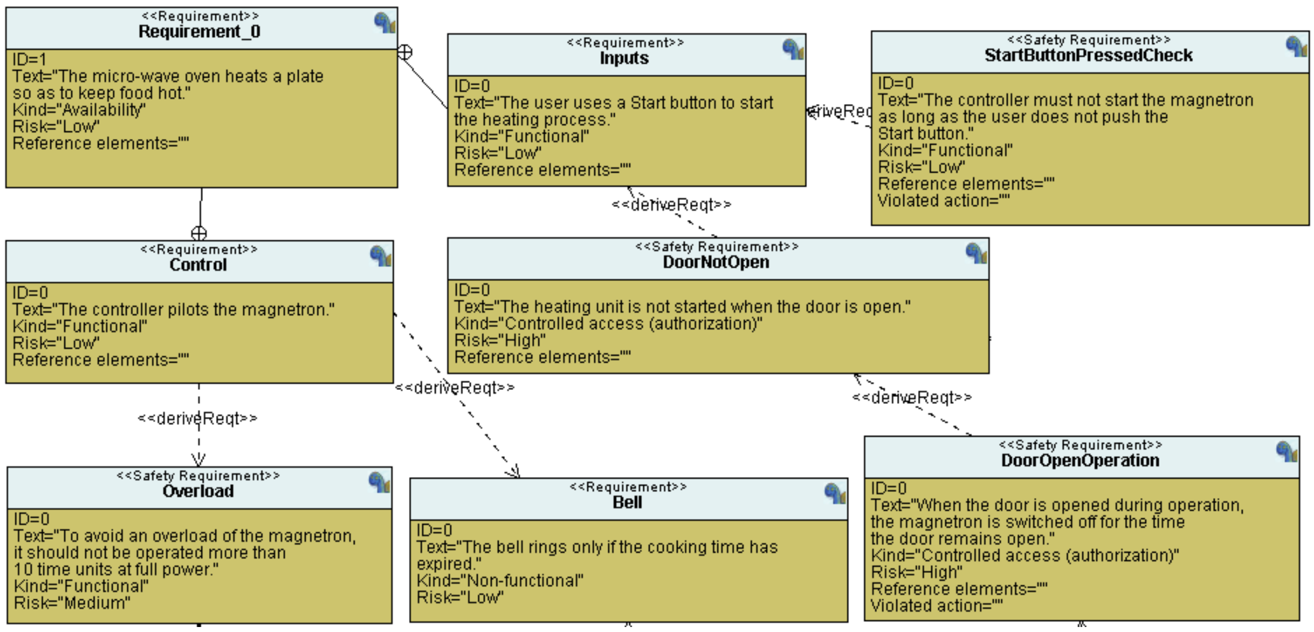
TTool allows for the capture of requirements, as per SysML. Some of these requirements may however be stereotyped as safety and security requirements, possibly derived from functional requirements. See Figure 12 for

[19] This constraint is a bit artificial in the case of a microwave oven, but it allows for the coverage of confidentiality, an important security criterion.

examples of safety requirements, and cf. Figure 13 for examples of security requirements. These requirements are explicitly shown here because they will be used throughout the running example.

In addition to the standard SysML tags (i.e. *id* and *text*) on requirements (OMG SysML, 2012), the tags defined on a standard AVATAR requirement are:

- Type, whose possible (fixed set) values are:
  - Functional,
  - Non-Functional,
  - Performance,
  - Privacy,
  - Confidentiality,
  - Non-repudiation,
  - Controlled access (authorization),
  - Availability,
  - Immunity,
  - Integrity,
  - Data origin authenticity,
  - Freshness,
  - Other;
- *Risk*, whose possible (fixed set) values are:
  - Low,
  - Medium,
  - High;
- *References*.


<<Requirement>>
Requirement_0
ID=1
Text="The micro-wave oven heats a plate so as to keep food hot."
Kind="Availability"
Risk="Low"
Reference elements=""

<<Requirement>>
Inputs
ID=0
Text="The user uses a Start button to start the heating process."
Kind="Functional"
Risk="Low"
Reference elements=""

<<Safety Requirement>>
StartButtonPressedCheck
ID=0
Text="The controller must not start the magnetron as long as the user does not push the Start button."
Kind="Functional"
Risk="Low"
Reference elements=""
Violated action=""

<<Requirement>>
Control
ID=0
Text="The controller pilots the magnetron."
Kind="Functional"
Risk="Low"
Reference elements=""

<<Safety Requirement>>
DoorNotOpen
ID=0
Text="The heating unit is not started when the door is open."
Kind="Controlled access (authorization)"
Risk="High"
Reference elements=""

<<Safety Requirement>>
Overload
ID=0
Text="To avoid an overload of the magnetron, it should not be operated more than 10 time units at full power."
Kind="Functional"
Risk="Medium"

<<Requirement>>
Bell
ID=0
Text="The bell rings only if the cooking time has expired."
Kind="Non-functional"
Risk="Low"

<<Safety Requirement>>
DoorOpenOperation
ID=0
Text="When the door is opened during operation, the magnetron is switched off for the time the door remains open."
Kind="Controlled access (authorization)"
Risk="High"
Reference elements=""
Violated action=""

Figure 12: AVATAR requirements diagram for the microwave system (extract with safety requirements)

When the requirement is stereotyped as a safety requirement, an additional tag called *Violated action* is defined. Because safety requirements are verified through the use of observers (cf. §4.3.5.2 and §4.3.6.6), it is possible to document here which action of a model observer will be triggered when the safety requirement is violated.

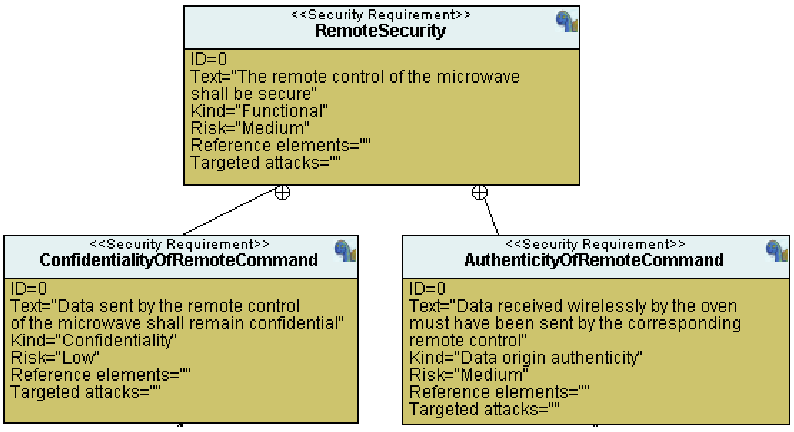

<<Security Requirement>>
RemoteSecurity
ID=0
Text="The remote control of the microwave shall be secure"
Kind="Functional"
Risk="Medium"
Reference elements=""
Targeted attacks=""
<<Security Requirement>>
ConfidentialityOfRemoteCommand
ID=0
Text="Data sent by the remote control of the microwave shall remain confidential"
Kind="Confidentiality"
Risk="Low"
Reference elements=""
Targeted attacks=""
<<Security Requirement>>
AuthenticityOfRemoteCommand
ID=0
Text="Data received wirelessly by the oven must have been sent by the corresponding remote control"
Kind="Data origin authenticity"
Risk="Medium"
Reference elements=""
Targeted attacks=""

Figure 13: AVATAR requirements diagram for the microwave system (extract with security requirements)

When the requirement is stereotyped as a security requirement, an additional tag called *targeted attacks* (cf. Figure 13) is defined. The latter allows for the tracing of the requirement to an *attack tree node*, as described in §4.3.3. The attribute has been implemented to allow for an automated check of the coverage of attacks, but, at the date of writing this report, the verification itself has not yet been implemented. Moreover the label of the attack tree node is currently entered textually, and there is no syntax verification that this attack tree node really exists. The implementation of the automated check of the coverage of attacks is in TTool's improvement roadmap.

Assessment take-away

The AVATAR extension to SysML requirements seems interesting.

However, the full potential of the extensions cannot actually be exploited because the necessary analysis functions have not (yet) been implemented.

4.3.3 System analysis

To support system analysis, TTool implements:

- use cases diagrams, cf. Figure 42;
- context diagrams, cf. Figure 43;
- sequence diagrams, cf. Figure 44;
- activity diagrams, cf. Figure 45;
- and attack tree diagrams, cf. Figure 14.

Since the first four types of diagrams are not specific to safety and security engineering, we did not assess AVATAR with respect to these diagrams, and some illustrations are only given in Annex A – Microwave Use Case Supplementary Data on page 56. Thus, we present here, as part of system analysis, only the attack tree diagram of AVATAR (Apvrille, et al., 2014).

Instead of using a traditional attack tree approach, the AVATAR methodology suggests that threats can be modelled with a more relational approach, using slightly customized SysML parametric diagrams. Threats are modelled as values embedded into blocks that represent the target of the attacks, thus achieving a representation that is more compact and better mapped to the system architecture. Attacks (i.e. artefact stereotyped with << attack >>) can be linked together with logical operators, like *OR*, and *AND*, as well as temporal causality operators, like *SEQUENCE*, *BEFORE*, or *AFTER*. The latter constructs are helpful to describe situations of real-time embedded systems in which there is a maximum duration between two causally related attacks, e.g. due to the expiration of a time-limited authentication.

As mentioned in §4.3.2, security requirements can be linked to attacks.

Figure 14 shows an attack tree for the flashing process of the microwave oven running example. It is assume that the flashing can be done remotely (i.e. via Internet), or by a maintenance station.

The documentation provided herein on attack trees is rather short as the approach is new and still under development as part of the SysML-sec project at Telecom Paris-Tech. Further publications are expected soon (Apvrille, et al., 2015).

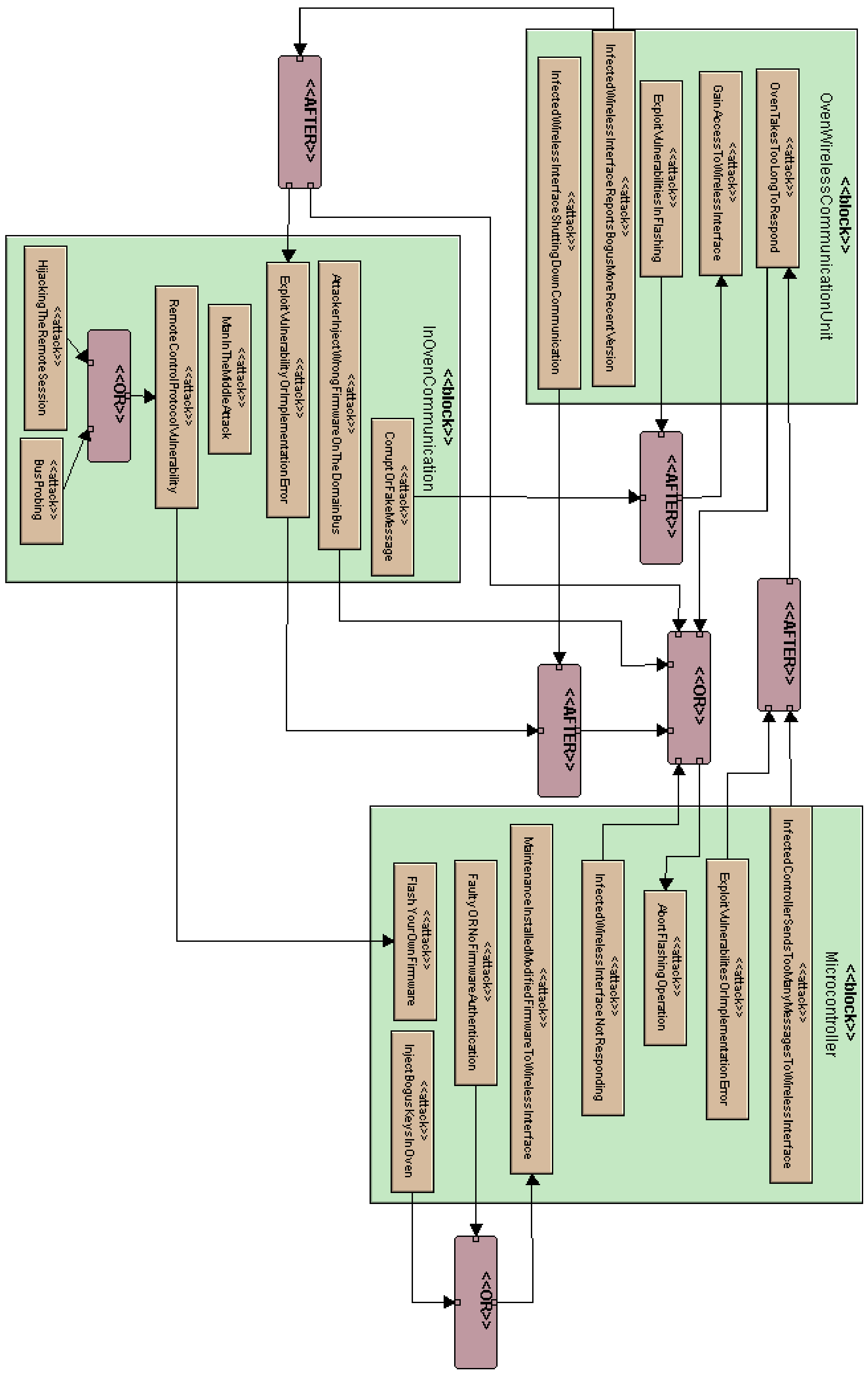

<<block>>
OvenWirelessCommunicationUnit
<<attack>>
OvenTakesTooLongToRespond
<<attack>>
GainAccessToWirelessInterface
<<attack>>
ExploitVulnerabilitiesInFlashing
<<attack>>
InfectedWirelessInterfaceReportsBogusMoreRecentVersion
<<attack>>
InfectedWirelessInterfaceShuttingDownCommunication
<<AFTER>>
<<block>>
InOvenCommunication
<<attack>>
CorruptOrFakeMessage
<<attack>>
AttackerInjectWrongFirmwareOnTheDomainBus
<<attack>>
ExploitVulnerabilityOrImplementationError
<<attack>>
ManInTheMiddleAttack
<<attack>>
RemoteControlProtocolVulnerability
<<OR>>
<<attack>>
HijackingTheRemoteSession
<<attack>>
BusProbing
<<AFTER>>
<<AFTER>>
<<OR>>
<<AFTER>>
<<block>>
Microcontroller
<<attack>>
InfectedControllerSendsTooManyMessagesToWirelessInterface
<<attack>>
ExploitVulnerabilitiesOrImplementationError
<<attack>>
AbortFlashingOperation
<<attack>>
InfectedWirelessInterfaceNotResponding
<<attack>>
MaintenanceInstalledModifiedFirmwareToWirelessInterface
<<attack>>
FaultyORNoFirmwareAuthentication
<<attack>>
FlashYourOwnFirmware
<<attack>>
InjectBogusKeysInOven
<<OR>>

Figure 14: Example of AVATAR attack tree

A weakness of the AVATAR implementation is the fact that there is no consistency check between the SysML blocks defined in the context diagrams, the attack tree diagrams and the design block diagrams.

A weakness of the AVATAR methodology is the fact that attack tree diagrams may be defined only after a first system design has been performed. Thus, some form of loop should be enacted between the *Analysis* and the *Design* phases of the methodology.

Assessment take-away

The AVATAR version of the attack tree is an innovative relational approach. It needs to be assessed with respect to scalability.

4.3.4 System design

System design is the phase of the system / software life-cycle for which AVATAR brings the most added-value with respect to safety (Apvrille, et al., 2013) and security (Pedroza, et al., 2011). This assessment section is therefore rather long and organised per major AVATAR feature. The main diagram for the AVATAR system design is the block diagram, with the automatic creation of a state-machine diagram per block defined in the block diagram.

In line with SysML, an AVATAR block defines a list of attributes, methods and signals. Signals can be sent over synchronous or asynchronous channels[20]. Channels are defined using connectors between ports. Those connectors contain a list of signal associations.

AVATAR state machine diagrams are built upon SysML state machines, including hierarchical states, but enhanced with temporal operators to deal with task complexity and delay between tasks.

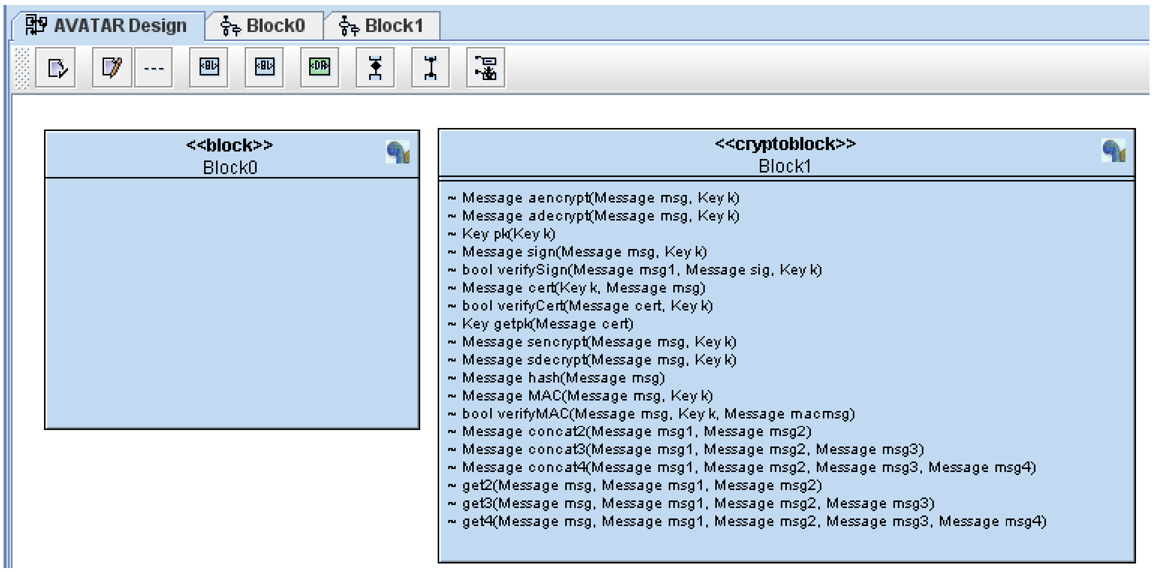
This section only describes enhancements with respect to standard SysML diagrams.

4.3.4.1 Defining blocks to handle secured communications

Handling secured communications is a common requirement for security engineering, and indeed, our case study has one such requirement (cf. Figure 13). AVATAR provides specific support for this.

The AVATAR block diagram allows for the creation of two types of blocks (cf. Figure 15):

- standard SysML blocks, stereotyped <<*block*>> in accordance to the SysML standard (OMG SysML, 2012);
- blocks that need to handle secured communications, stereotyped <<*cryptoblock*>>.



<<block>>
Block0

<<cryptoblock>>
Block1

```
~ Message aencrypt(Message msg, Key k)
~ Message adecrypt(Message msg, Key k)
~ Key pk(Key k)
~ Message sign(Message msg, Key k)
~ bool verifySign(Message msg1, Message sig, Key k)
~ Message cert(Key k, Message msg)
~ bool verifyCert(Message cert, Key k)
~ Key getpk(Message cert)
~ Message sencrypt(Message msg, Key k)
~ Message sdecrypt(Message msg, Key k)
~ Message hash(Message msg)
~ Message MAC(Message msg, Key k)
~ bool verifyMAC(Message msg, Key k, Message macmsg)
~ Message concat2(Message msg1, Message msg2)
~ Message concat3(Message msg1, Message msg2, Message msg3)
~ Message concat4(Message msg1, Message msg2, Message msg3, Message msg4)
~ get2(Message msg, Message msg1, Message msg2)
~ get3(Message msg, Message msg1, Message msg2, Message msg3)
~ get4(Message msg, Message msg1, Message msg2, Message msg3, Message msg4)
```

Figure 15: AVATAR blocks and cryptoblocks

The crypto block presupposes the existence of two data types called *Key* and *Message* (cf. Figure 16). These data types must be defined by the designer.

[20] From the ProVerif viewpoint, successive flows on a channel are not ordered; for its proof, ProVerif will verify all sequencing options, possibly leading to a combinatorial explosion. The designer must be careful not to multiply flows if they can be concatenated within a unique flow.

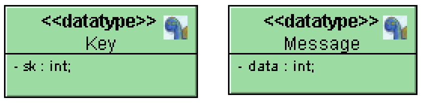


Figure 16: Data types required for the AVATAR cryptoblocks

Using these two data types, the cryptoblock defines 19 methods (with a package visibility) that are commonly used to handle secured communications:

- *aencrypt(Message msg, Key pub)* and a*decrypt(Message msg, Key priv)*, for respectively encrypting and decrypting messages with asymmetric keys;
- *sencrypt(Message msg, Key k)* and *sdecrypt(Message msg, Key k)*, for respectively encrypting and decrypting messages with a symmetric key;
- *MAC(Message msg, Key k)* and *verifyMAC(Message msg, Key k, Message macm)*, for respectively computing and verifying message authentication codes; here, the model is very strong: it considers the MAC essentially as a random oracle[21], which is much stronger than the typical computational assumption on MACs (unforgeability);
- *sign(Message msg, Key priv)* and *verifySign(Message msg, Message sig, Key pub),* for respectively signing and verifying the signature of a message;
- *hash(Message msg),* for generating a hash ; it takes as input and returns a message; it captures pre-image resistance[22], second pre-image resistance[23] and collision resistance[24] properties;
- *pk(Key k),* which takes an argument of type private key and returns a public key, to capture the notion of constructing a key pair; in §4.3.4.2, a pragma is defined to support the pairing of keys prior to system execution; the pk() method supports dynamic pairing during system execution;
- *cert(Key pub, Message caSignedPub)*, *verifyCert(Message cert, Key caPub)* and *getpk(Message cert)* for respectively creating a cryptographic certificate[25], verifying a cryptographic certificate and extracting the public key from the cryptographic certificate;
- *concat2(Message msg1, Message msg2), concat3(Message msg1, Message msg2, Message msg3)* , *concat4(Message msg1, Message msg2, Message msg3, Message msg4)* and *get2(Message msg, Message msg1, Message msg2), get3(Message msg, Message msg1, Message msg2, Message msg3)* and *get4(Message msg, Message msg1, Message msg2, Message msg3, Message msg4)* for concatenating messages, and reversely, extracting a message from a concatenated message.

The *concat* and *get* functions are rendered necessary by the fact that ProVerif supports only one datum at a time in a communication channel. The *concat* functions return an object of type Message. The *get* functions allow retrieving the multiple instances of messages that were concatenated using a *concat* function.

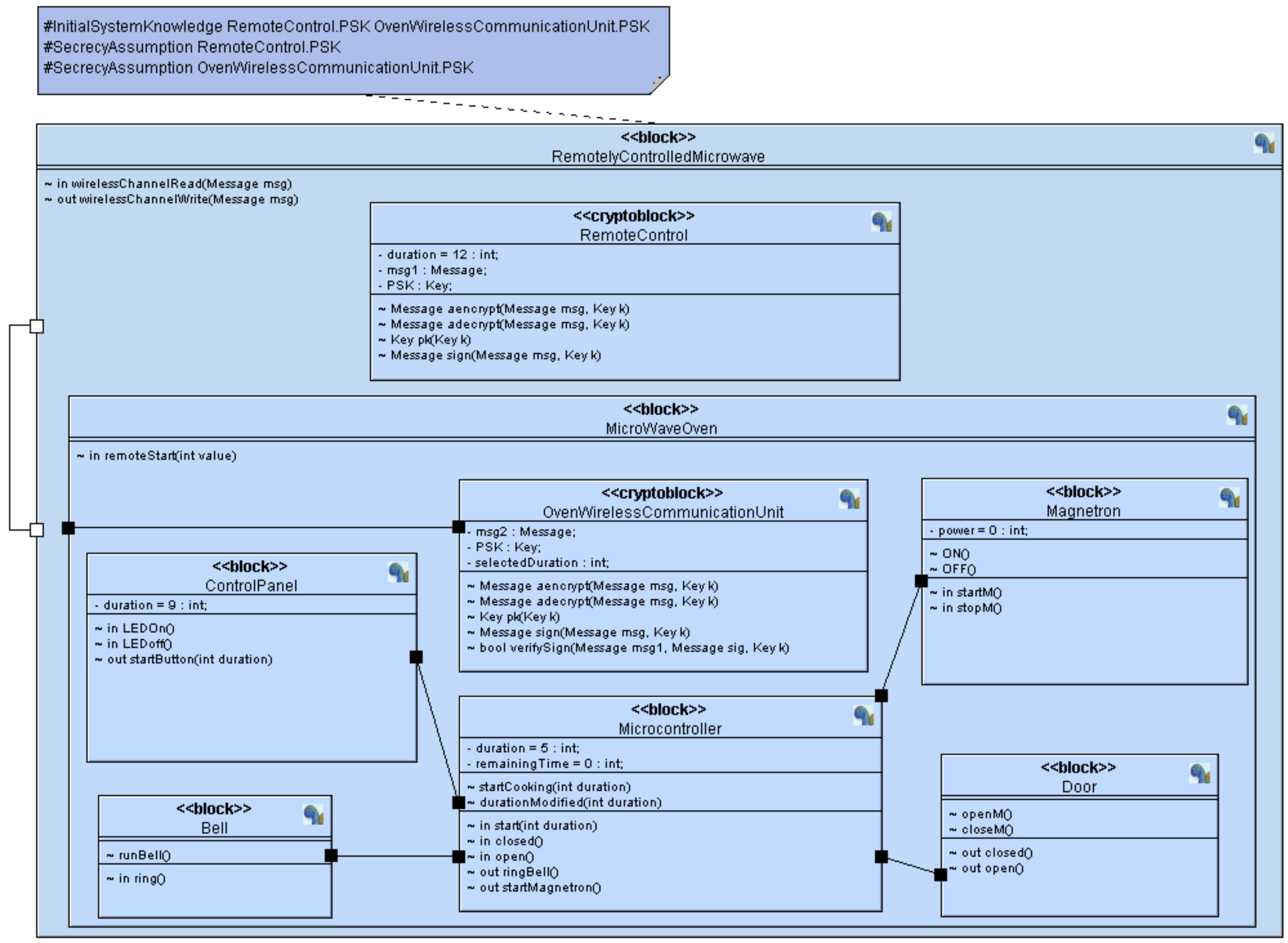
It is important to note that the use of the <<*cryptoblock*>> is not necessary to handle secured communications using AVATAR. It is only shorthand to declare usual cryptographic functions. The 19 methods predefined on a cryptoblock may be redefined by the designer on a standard SysML block. If the method signatures are preserved, the methods will be recognised for what they are, as defined above.

The microwave oven design is made using several types of blocks and elements (see Figure 17):

- a main block named *RemotelyControlledMicrowave*, which contains all other blocks modelling the system, i.e. the *RemoteControl*, and the *MicroWaveOven, which is* itself composed of a *OvenWirelessComminucationUnit*, a *Microcontroller*, a *Magnetron*, a *Door*, a *Bell* and a *ControlPanel*; each block declares attributes, methods and signals;
- the declaration of two data types (i.e. *Key* and *Message*);
- the declaration of communication channels[26] between blocks;
- the declaration of a security-related constraint in the note located at the top of the diagram.

[21] I.e., a theoretical black box that responds to every unique query with a truly random response chosen uniformly from its output domain.
[22] I.e., for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output.
[23] I.e., it is computationally infeasible to find any second input that has the same output as a specified input.
[24] I.e., it is computationally infeasible to find any two distinct inputs x, x′ which hash to the same output.
[25] A cryptographic certificate is a user's public key, which has been signed and encrypted using the private key of a well-known Certificate Authority.
[26] Ports filled in black represent synchronous communication whereas ports filled in white represent asynchronous communications. Signals and ports can be used by the block declaring them, and by the blocks it contains. For example, all blocks may use the asynchronous channel connecting RemotelyControlledMicrowave to itself.

Note here that the *RemoteControl* and the *OvenWirelessCommunicationUnit* have been declared as cryptoblocks.



```
#InitialSystemKnowledge RemoteControl.PSK OvenWirelessCommunicationUnit.PSK
#SecrecyAssumption RemoteControl.PSK
#SecrecyAssumption OvenWirelessCommunicationUnit.PSK
```

<<block>> RemotelyControlledMicrowave
~ in wirelessChannelRead(Message msg)
~ out wirelessChannelWrite(Message msg)
<<cryptoblock>> RemoteControl
- duration = 12 : int;
- msg1 : Message;
- PSK : Key;
~ Message aencrypt(Message msg, Key k)
~ Message adecrypt(Message msg, Key k)
~ Key pk(Key k)
~ Message sign(Message msg, Key k)
<<block>> MicroWaveOven
~ in remoteStart(int value)
<<cryptoblock>> OvenWirelessCommunicationUnit
- msg2 : Message;
- PSK : Key;
- selectedDuration : int;
~ Message aencrypt(Message msg, Key k)
~ Message adecrypt(Message msg, Key k)
~ Key pk(Key k)
~ Message sign(Message msg, Key k)
~ bool verifySign(Message msg1, Message sig, Key k)
<<block>> Magnetron
- power = 0 : int;
~ ON()
~ OFF()
~ in startM()
~ in stopM()
<<block>> ControlPanel
- duration = 9 : int;
~ in LEDOn()
~ in LEDoff()
~ out startButton(int duration)
<<block>> Microcontroller
- duration = 5 : int;
- remainingTime = 0 : int;
~ startCooking(int duration)
~ durationModified(int duration)
~ in start(int duration)
~ in closed()
~ in open()
~ out ringBell()
~ out startMagnetron()
<<block>> Bell
~ runBell()
~ in ring()
<<block>> Door
~ openM()
~ closeM()
~ out closed()
~ out open()

Figure 17: Overall AVATAR Block Diagram for the Microwave System

A minor weakness of the AVATAR implementation is the fact that blocks cannot be refactored, for example to change their stereotypes from <<*block*>> to <<*cryptoblock*>>.

Assessment take-away

The AVATAR cryptoblock extension to SysML blocks is a key feature of AVATAR with respect to security engineering: it is both easy to use and highly productive to design secure communications.

4.3.4.2 Modelling cryptographic keys

This section explains how symmetric and asymmetric keys can be declared.

Pre-sharing of symmetric keys between explicitly stated actors

When symmetric keys are used, it is important to capture the fact that the keys are confidential and pre-shared between the communicating actors prior to the communication itself.

SysML offers several ways to share data between classes, using for example block attributes, or using a dedicated block storing shared knowledge. Unfortunately, those solutions suffer from two drawbacks: (i) the sharing is not really explicit, i.e., it is not clear which block intends to use the shared data; (ii) the sharing is defined for the entire system execution. To overcome those two limitations, AVATAR proposes to use specific directives - or pragmas - in notes of block diagrams. The syntax of the two pragmas is the following:

```
#InitialSystemKnowledge BlockID.attributeID [BlockID.attributeID]*
#InitialSessionKnowledge BlockID.attributeID [BlockID.attributeID]*
```

The pragmas specify that the values of the listed attributes are identical and shared respectively prior[27] to the use of the system, or the running of a session.

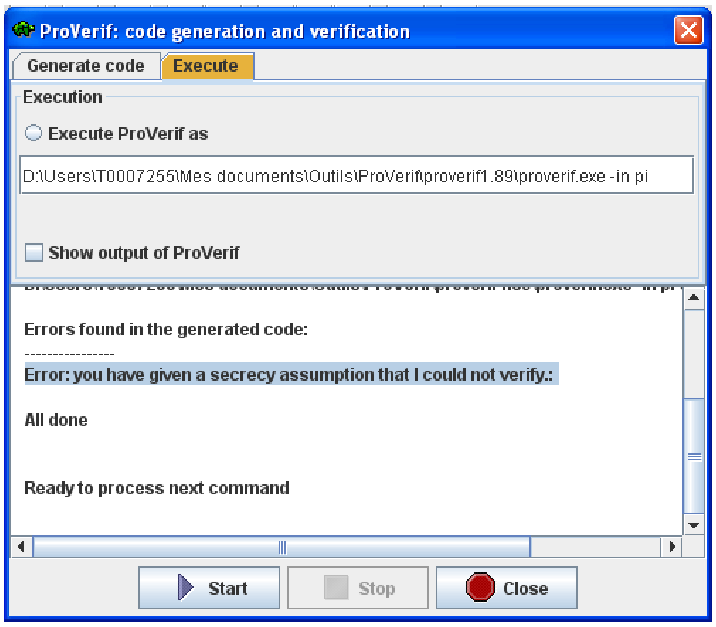
In our running example (cf. Figure 17), the assumption "*The wireless interface of the microwave oven and its remote control share a common symmetric encryption key*" (cf. §4.3.1) is captured as follows:

```
#InitialSystemKnowledge RemoteControl.PSK OvenWirelessCommunicationUnit.PSK
```

In addition, to cover the confidential nature of the key, AVATAR proposes another pragma, whose syntax is as follows:

```
#SecrecyAssumption BlockID.attributeID
```

According to (Blanchet, et al., 2014), this pragma formally asserts that the attacker cannot have access to the key. This assumption differs from the informal assumptions discussed in §4.3.1 in that this assertion is checked by ProVerif, thus preserving soundness. It also differs from the confidentiality property (cf. §4.3.5.1) in that, if the claim is true, no feedback is given by the tool, and if the claim is false, the verification is stopped prior to the verification of any other property (cf. Figure 18).



```
D:\Users\T0007255\Mes documents\Outils\ProVerif\proverif1.89\proverif.exe -in pi
```

```
Errors found in the generated code:
----------------
Error: you have given a secrecy assumption that I could not verify.:

All done


Ready to process next command
```

Figure 18: Secrecy assumption verification error

This syntax, based on pragmas, correctly covers the modelling need for the pre-sharing of confidential symmetric cryptographic keys. However, a small inconvenience of this approach is that traceability of security requirements to the pragmas cannot be ensured, because UML notes do not have identifiers.

Pairing public and private keys

When asymmetric keys are used, it is important to model the fact that: (i) the public and private keys are paired; (ii) the public key is indeed public, and therefore known to the attacker and all communicating parties; (iii) the private key is indeed secret. To cover all these modelling needs for key pairing before system execution, AVATAR proposes to use a set of three pragmas:

```
#PrivatePublicKeys BlockID privKeyAttributeID pubKeyAttributeID
#InitialSystemKnowledge BlockID.attributeID [BlockID.attributeID]*
#SecrecyAssumption BlockID.attributeID
```

The first pragma declares the pairing of the public and private keys. With respect to the attacker model of ProVerif, this pragma also implicitly publishes the public key to the attacker.

The second and third pragmas are identical to the ones used for symmetric keys: they simply allow declaring that the communicating parties all store a copy of the same public key, and that all these copies are confidential.

An example of static key pairing, i.e. pairing before system execution, is given in Figure 19. In case dynamic paring (i.e. pairing during system execution) is required, the pk() function can be used, cf. §4.3.4.1 for more details.

[27] For the designer, it is important to note that these attributes must not be assigned a value at runtime, or else the shared value property is considered lost.

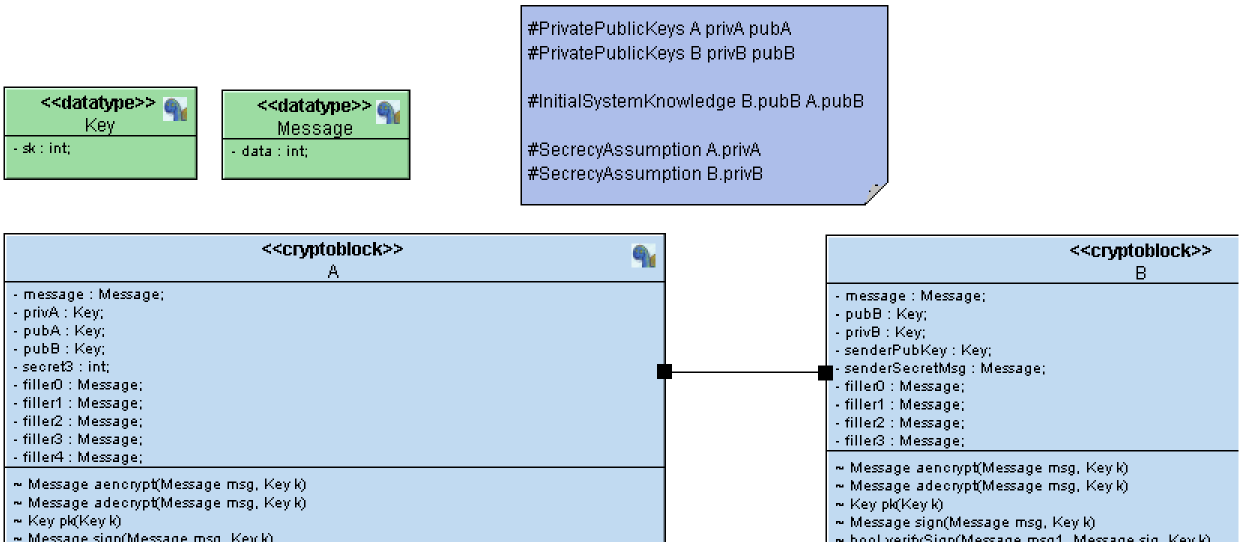


Figure 19: Example of static key pairing

As for the symmetric keys, this syntax correctly covers the modelling need, but forbids traceability of security requirements to the pragmas.

About certificates

AVATAR does not allow for the modelling of initial knowledge concerning certificates. This means that when a system is assumed to have one or more pre-installed certificates, then the complete installation process involving the certification authority must be designed using state-machines.

Assessment take-away

The AVATAR extensions to model cryptographic keys are concise, expressive and readable with respect to defining different types of keys.

AVATAR is a significantly weaker concerning the modelling of certificates, since certificates can only be modelled dynamically (i.e. no initial knowledge concerning certificates). This approach may become dramatically complex in case of certification chains.

4.3.4.3 Defining the real-time behaviour of blocks using temporal operators in state-machines

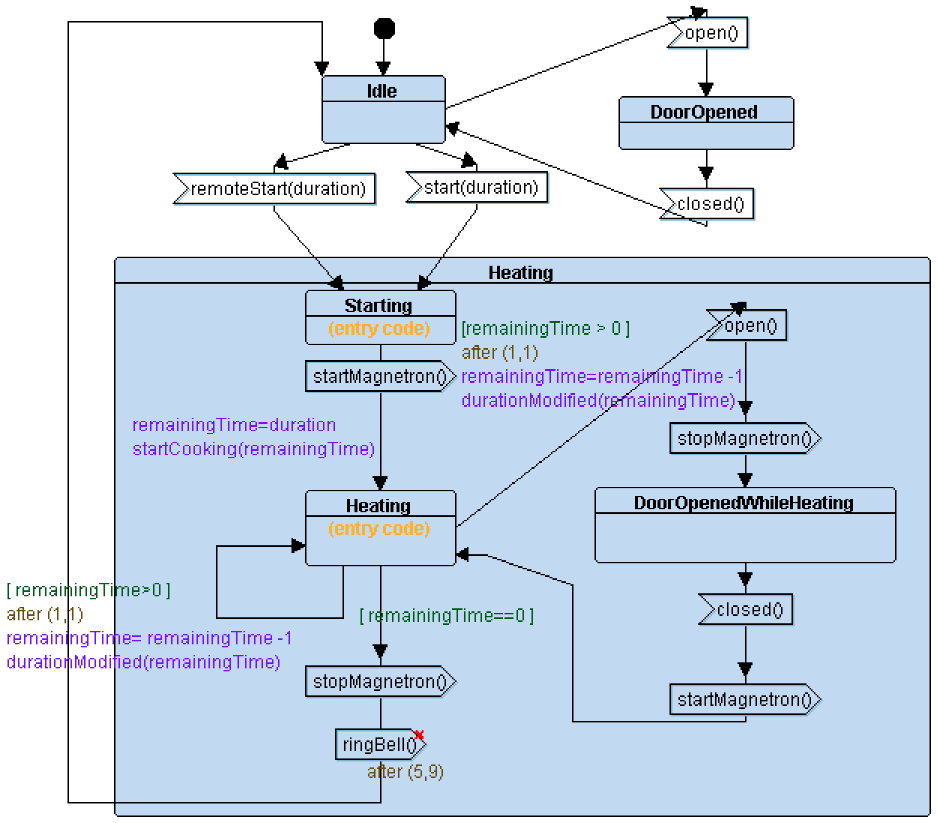


Figure 20: AVATAR block state-machine of the microwave oven controller

Each time a block is created, TTool creates an empty state-machine diagram for that block. AVATAR state machine diagrams are built upon SysML state machines, enhanced with two temporal operators to support real-time system schedulability analysis:

- *after (tmin, tmax)*: it models a variable delay during which the activity of the block is suspended, waiting for a delay between *tmin* and *tmax* to expire;
- *computeFor (tmin, tmax)*: it models a time during which the activity of the block actively executes instructions, before transiting to the next state: that computation may last from *tmin* to *tmax* units of time.

The most important state-machine for the microwave oven running example is the state-machine of the Controller block; it is given in Figure 20. It shows multiple uses of the after() method.

Assessment take-away

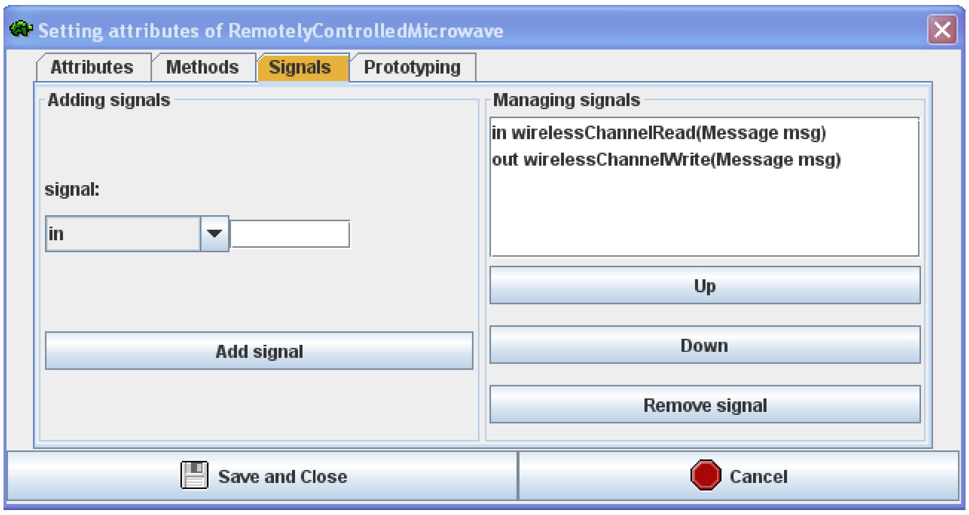
The temporal operators offered by AVATAR are very far from the capabilities of MARTE, but they offer the minimal operators to start defining the real-time behaviour of blocks.

4.3.4.4 Defining the communication architecture

With AVATAR, the goal of the communication architecture is only[28] to support the application's control logic. This can (and must) be done using simple types (i.e. *Int*, *Bool*, and *Timer*) or user-defined data types, but the

[28] The application's algorithms, requiring more complex data types, must be written in C code, as part of the body of methods.

latter may themselves only be defined using integers and Booleans. Moreover, if the intent is to use ProVerif, the communications must be limited to a single datum element per communication channel flow[29].



Setting attributes of RemotelyControlledMicrowave

Attributes | Methods | Signals | Prototyping

Adding signals

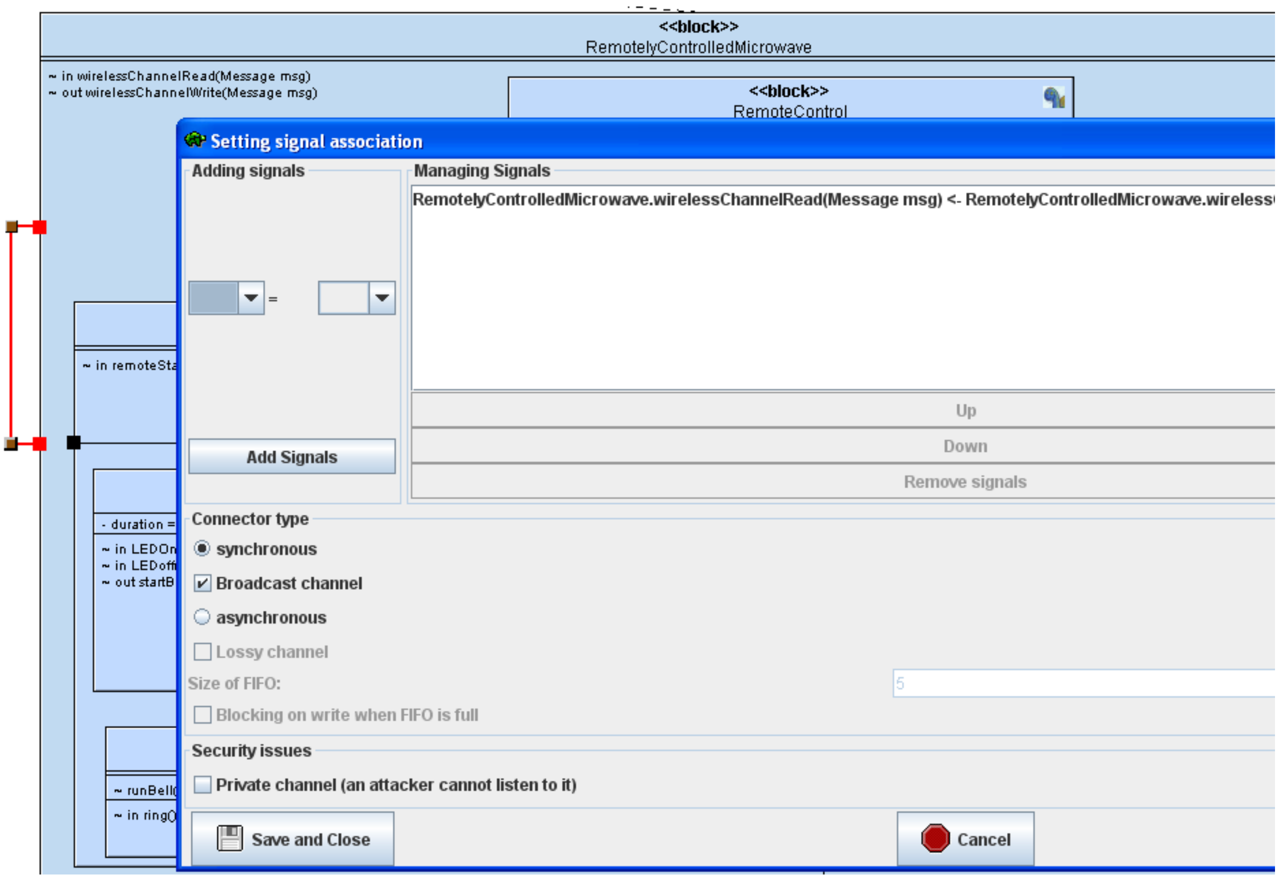
signal: in

Managing signals

in wirelessChannelRead(Message msg)
out wirelessChannelWrite(Message msg)

Figure 21: Declaration of Flow Properties of type Signal

The communication architecture is designed with flow properties of type Signal. An "out" FlowProperty of type Signal means that the owning Block may send the signal via connectors and an "in" FlowProperty means that the owning block is able to receive the Signal (cf. Figure 21).



<<block>> RemotelyControlledMicrowave

~ in wirelessChannelRead(Message msg)
~ out wirelessChannelWrite(Message msg)

<<block>> RemoteControl

Setting signal association

Adding signals

Managing Signals

RemotelyControlledMicrowave.wirelessChannelRead(Message msg) <- RemotelyControlledMicrowave.wireless

Connector type

● synchronous

[✓] Broadcast channel

○ asynchronous

☐ Lossy channel

Size of FIFO: 5

☐ Blocking on write when FIFO is full

Security issues

☐ Private channel (an attacker cannot listen to it)

Figure 22: Configuring communication channels with respect to security

The association between "in" and "out" signals is performed through the configuration of port connectors, cf. Figure 22. Beyond the usual configuration, e.g. synchronous vs. asynchronous communication, the AVATAR settings include a toggle switch specifying if the connector represents a private channel, i.e. a channel that an attacker cannot listen to.

[29] For complex flows, see the get() and concat() methods in §4.3.4.1.

In our running example, a synchronous broadcast public connection is configured between the remote control and the oven. Connections between elements inside the oven are considered private.

TTool/AVATAR integrates, through the use of ProVerif (Blanchet), its own attacker model, which is a process implementing a Dolev-Yao approach (Dolev, et al., 1983). In other words, the attacker model is implicit, i.e., there is no need to model an attacker either at block diagram level, or at state machine diagram level. The process implementing the Dolev-Yao approach acts like an adversary relying upon a set of known names, variables and terms which is referred to as knowledge. The increase of the attacker knowledge relies on public channel probing and execution of functions that are non-prohibited to the attacker. The attacker is assumed to be an active eavesdropper, i.e. someone who: (i) can obtain any message passing through public networks; (ii) is a legitimate user of the network, i.e. can initiate a conversation with any other user; (iii) has the opportunity to be a receiver to any other user. More details are given in (Dolev, et al., 1983) and (Pedroza, et al., 2011).

Assessment take-away

Overall, the AVATAR extensions to SysML for the communication architecture are deemed useful and easy to configure.

4.3.5 Property modelling

This section should normally corresponds to the Property Modelling phase of the AVATAR methodology (as described in §4.2), which is dedicated to the modelling of safety properties using the TEPE language. However, the transformation of TEPE towards UPPAAL has not (yet) been implemented, thus rendering the formalisation somehow useless, because automated verification cannot be performed. This section therefore focuses on the specification of security properties (cf. §4.3.5.1) and safety properties (cf. §4.3.5.10) that can be formally verified. The reader interested in the TEPE language and the way to formalise safety properties using TEPE may refer to Annex B and Annex C.

4.3.5.1 Modelling of security properties

Security properties can usually be defined with a criterion (e.g., confidentiality), and with a few elements related to that criterion (e.g., the confidentiality of an attribute of a block). In AVATAR, that simplicity results in a simple modelling solution relying on pragmas provided in notes of block diagrams.

Confidentiality

Confidentiality is the assurance that information is not disclosed to system entities (users, processes, devices) unless they have been authorized to access the information. With AVATAR, the confidentiality of an attribute of a block is modelled as a simple pragma provided in the note of a block diagram, according to the following syntax:

```
#Confidentiality BlockID.attributeID
```

In our running example, we have a confidentiality requirement (cf. Figure 13), stating that "*Data sent by the remote control of the microwave shall remain confidential*". The corresponding property in AVATAR is illustrated in Figure 23.

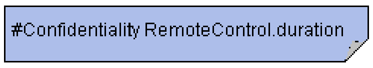


Figure 23: Example of confidentiality property

Details on the verification of a confidentiality property is given in §4.3.6.2.

Authenticity

Authenticity is confidence in the validity of a transmission, a message, or message originator. In AVATAR, the authenticity of a message transmission between a block A and a block B is modelled as a pragma that states that a message m2 received by the block B was necessarily sent before in a message m1 by the block A. The syntax of the authenticity pragma requires the specification of two states:

- state s1, corresponding to the state of block A <u>right before</u>[30] the sending of m1,

[30] It is important du note that the message m1 must be formed before state s1, i.e. it is not allowed to form m1 on the transition between s1 and the sending of m1. If the message is formed between state s1 and the sending of m1, then the verification of the authenticity will pro-

- state s2, corresponding to the state of block B <u>after</u> message m2 has been received <u>and</u> accepted as authentic[31], hereby meaning that some certificate / signature verifications or decryption operation are normally performed[32].

The syntax is the following:

```
    #Authenticity SenderBlockId1.stateId1.messageId1 ReceiverBlockId2.stateId2.messageId2
```

In our running example, we have an authenticity requirement (cf. Figure 13) stating that "*Data received wirelessly by the oven must have been sent by the corresponding remote control*". The corresponding property in AVATAR can only be written after defining the state-machines (cf. Figure 24) of the communicating parties, i.e. the remote control and of the oven's wireless communication unit (cf. Figure 17).

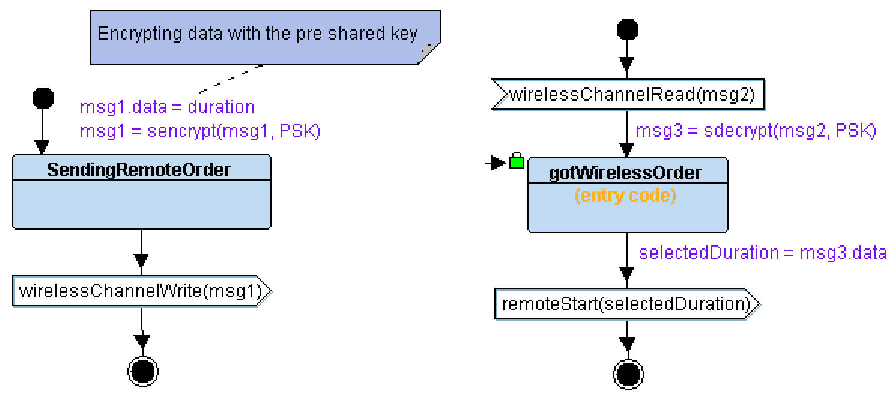
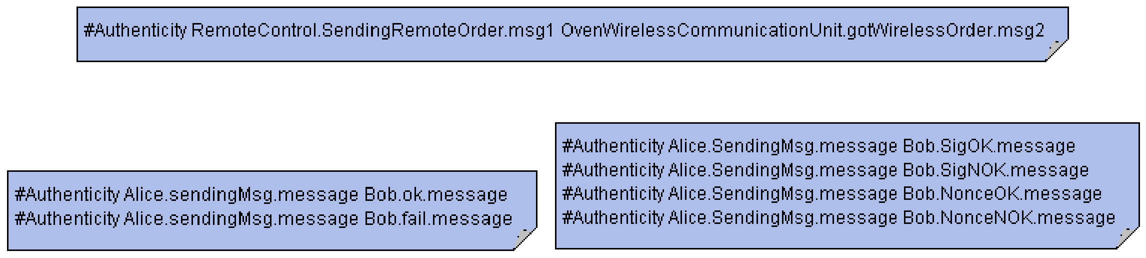

Encrypting data with the pre shared key
msg1.data = duration
msg1 = sencrypt(msg1, PSK)
SendingRemoteOrder
wirelessChannelWrite(msg1)
wirelessChannelRead(msg2)
msg3 = sdecrypt(msg2, PSK)
gotWirelessOrder
(entry code)
selectedDuration = msg3.data
remoteStart(selectedDuration)

Figure 24: State-machines of the remote control (left) and of the oven's wireless communication unit (right)

Given these two state-machines, the authenticity property of the message transmission can be expressed as shown in the first UML comment of Figure 25. The verification of this property should normally fail[33].



#Authenticity RemoteControl.SendingRemoteOrder.msg1 OvenWirelessCommunicationUnit.gotWirelessOrder.msg2

#Authenticity Alice.sendingMsg.message Bob.ok.message
#Authenticity Alice.sendingMsg.message Bob.fail.message

#Authenticity Alice.SendingMsg.message Bob.SigOK.message
#Authenticity Alice.SendingMsg.message Bob.SigNOK.message
#Authenticity Alice.SendingMsg.message Bob.NonceOK.message
#Authenticity Alice.SendingMsg.message Bob.NonceNOK.message

Figure 25: Examples of authenticity properties

We therefore built another simple example whereby Alice sends a signed message to Bob through a public channel; upon receiving the message, Bob checks the signature (cf. Figure 26). This protocol uses the sign() and verifySign() methods pre-defined on cryptoblocks, as explained in §4.3.4.1.

vide inconsistent results and no error message will be generated. The syntax is therefore error prone, and some form of syntax verification would be welcome.

[31] It is important to note that the signature verification or message decryption must be performed before state s2. If not, the verification of the authenticity will provide inconsistent results and no error message will be generated. Moreover, we discovered that the message variable used in the proof should not be re-used before state s2. For example, in the state-machine of Figure 24, writing "msg2=sdecrypt(msg2, PSK)" instead of "msg3 = sdecrypt(msg2, PSK)" confuses ProVerif and provides inconsistent results. The syntax is therefore error prone, and some form of syntax verification to avoid such pitfalls would be welcome.

[32] It is important to note that if a decryption operation is illegal, e.g. use of the wrong key, then the state-machine transition cannot be taken.

[33] In ProVerif, the formalization of symmetrical encryption is authenticated. It is indeed difficult to model general unauthenticated encryption in formal protocol provers. In reality, symmetrical encryption can support some level of authentication when the encrypted message is long: in that case, the semantics of the decrypted message allows the addressee to assess if the message was indeed encrypted with the symmetric key, or if it was a random message. By contrast, when the message is very short, e.g. an integer, any random value may be decrypted as an integer, thus allowing for impersonation attacks.

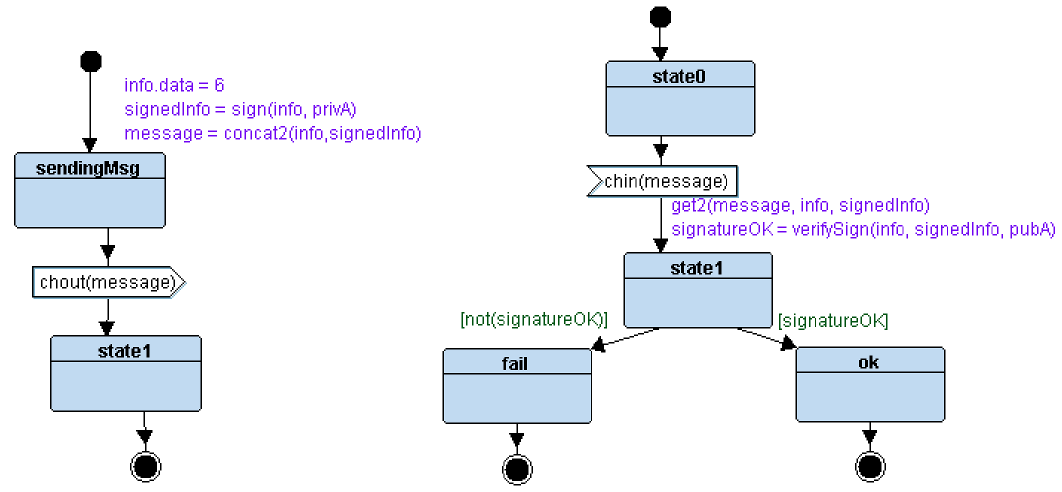

info.data = 6
signedInfo = sign(info, privA)
message = concat2(info,signedInfo)
sendingMsg
chout(message)
state1
state0
chin(message)
get2(message, info, signedInfo)
signatureOK = verifySign(info, signedInfo, pubA)
state1
[not(signatureOK)]
[signatureOK]
fail
ok

Figure 26: State-machines of Alice and Bob exchanging a signed message

Given these two state-machines, two authenticity properties of the message transmission were expressed, as shown in the second UML comment of Figure 25, to verify the AVATAR results depending on the validity of the signature.

However, this protocol is subject to replay.

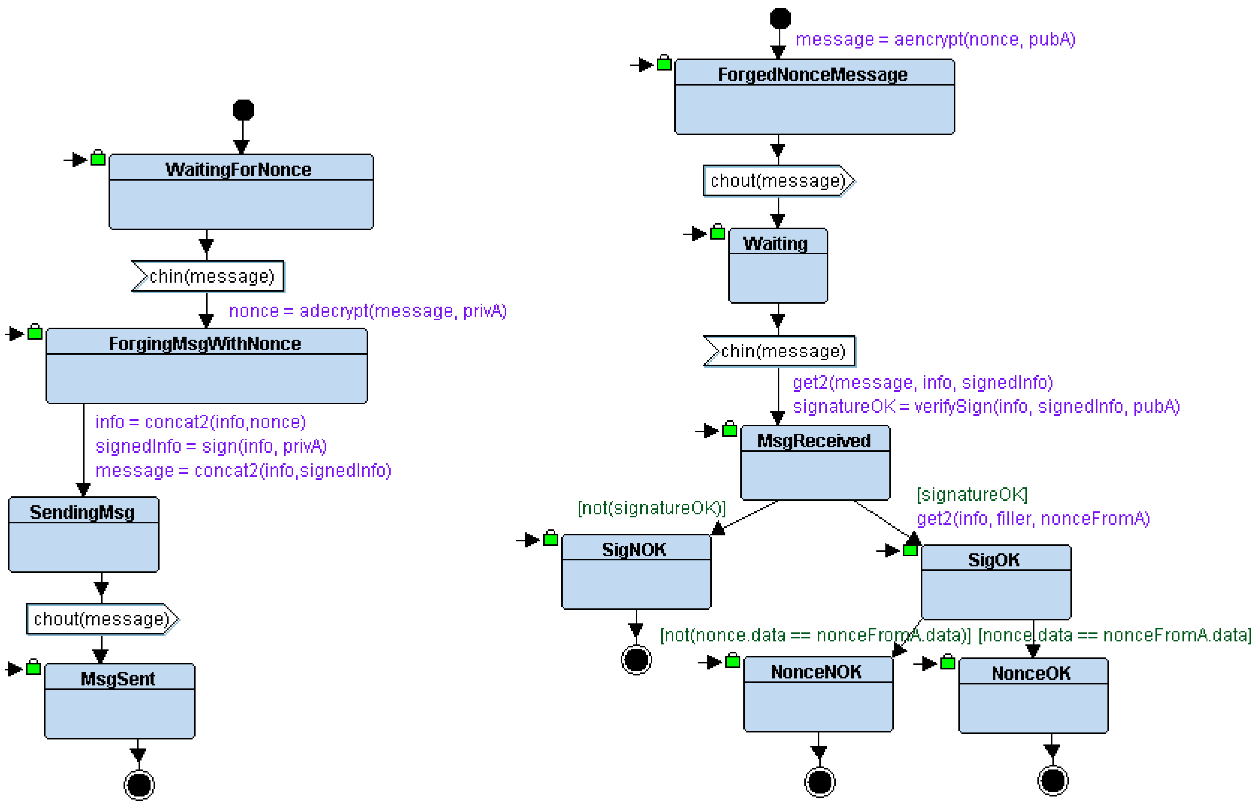

WaitingForNonce
chin(message)
nonce = adecrypt(message, privA)
ForgingMsgWithNonce
info = concat2(info,nonce)
signedInfo = sign(info, privA)
message = concat2(info,signedInfo)
SendingMsg
chout(message)
MsgSent
message = aencrypt(nonce, pubA)
ForgedNonceMessage
chout(message)
Waiting
chin(message)
get2(message, info, signedInfo)
signatureOK = verifySign(info, signedInfo, pubA)
MsgReceived
[not(signatureOK)]
[signatureOK]
get2(info, filler, nonceFromA)
SigNOK
SigOK
[not(nonce.data == nonceFromA.data)]
[nonce.data == nonceFromA.data]
NonceNOK
NonceOK

Figure 27: State-machines of Alice and Bob exchanging a signed message with nonce

Thus, we extended the Alice and Bob use case, to include a nonce. The corresponding state machines are given in Figure 27. Given these two state-machines, four authenticity properties of the message transmission were expressed, as shown in the last UML comment of Figure 25, to verify the AVATAR results depending on the validity of the signature, and the validity of the nonce.

The results of the verification of the authenticity properties are given in §4.3.6.5.

Assessment take-away

Overall, the AVATAR syntax for modelling confidentiality and authenticity is deemed both clear and user friendly.

However, as already pointed out, the use of pragmas for the modelling creates a traceability issue, as UML comments do not have identifiers.

4.3.5.2 Modelling of safety properties

To formally model safety properties, AVATAR offers to use observers, usually one observer per property. This is a second choice solution, because in UML / SysML, observers cannot observe events without consuming those events. Therefore, observers must be explicitly fed with events: this can only be done by modifying the state-machines of the blocks creating and / or consuming the original events, dramatically polluting the original system design block and state-machine diagrams[34].

Although it is purely a matter of interpretation[35], the literature distinguishes positive observers, which express possibility properties[36], from negative observers, which express safety properties[37].

To assess the capabilities of AVATAR in terms of safety property modelling, we will provide the formalisation of one safety requirements (cf. §4.3.2) of our microwave running example. The other properties can be built in a similar manner.

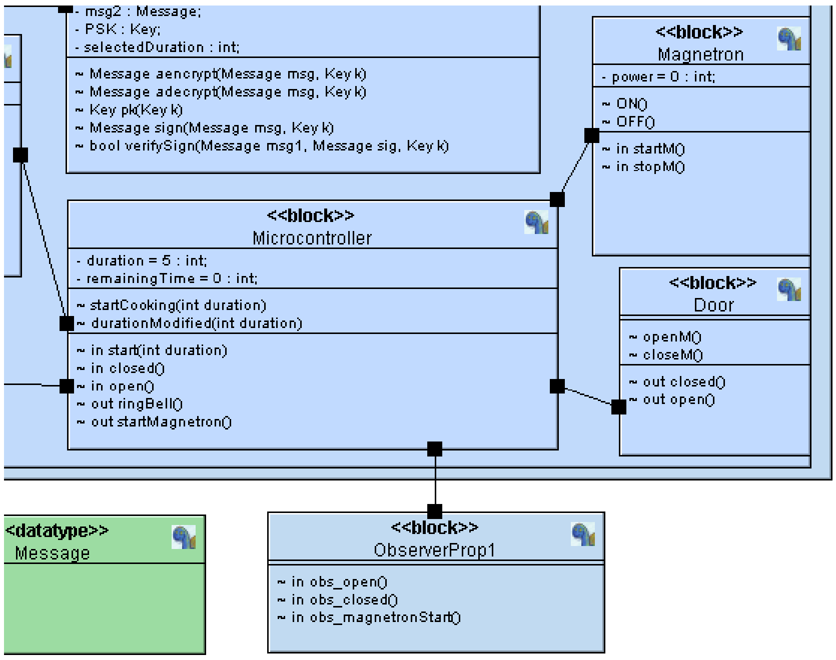


Figure 28: Observer for safety property n°1

The selected[38] safety requirement from our microwave running example is "*The heating unit is not started when the door is open.*"

To formalise the property on the basis of an observer, we must first create an observer block and make the assumption that it is capable of observing the opening and closing of the microwave's door, as well as the starting of the magnetron.

Considering the design of the microwave (cf. the state-machine of the *Microcontroller* in Figure 29 and the state-machines of the *Door* and the *Magnetron* in Figure 46), these three signals are accessible from the micro-

[34] This is why modelling properties with TEPE in the *Requirements Diagram* would have been a much cleaner approach, i.e. TEPE would automatically and transparently generate the observers. Unfortunately, safety properties formally modelled in TEPE cannot currently be formally verified with AVATAR (cf. introduction of §4.3.5.2), so observers must be manually crafted.

[35] Satisfying a negative observer amounts to violating a positive observer, and vice-versa.

[36] I.e., it is possible that something good will eventually happen (AG EF in CTL). This differs from liveness, i.e. something good will eventually happen (AF in CTL) and reachability (EF in CTL).

[37] I.e., nothing bad ever happens.

[38] This same example is also provided as a TEPE formalisation, cf. Annex C.

controller. It is thus possible[39] to connect the observer to the oven's microcontroller, and to define the corresponding events on the observer, as shown in Figure 28.

The state-machine of the *Microcontroller* in Figure 29 shows a modified state-machine in which all open, close and start signals are immediately repeated for the benefit of the observer. Looking at Figure 20, it can be compared to the same state-machine before adjunction of the observer.

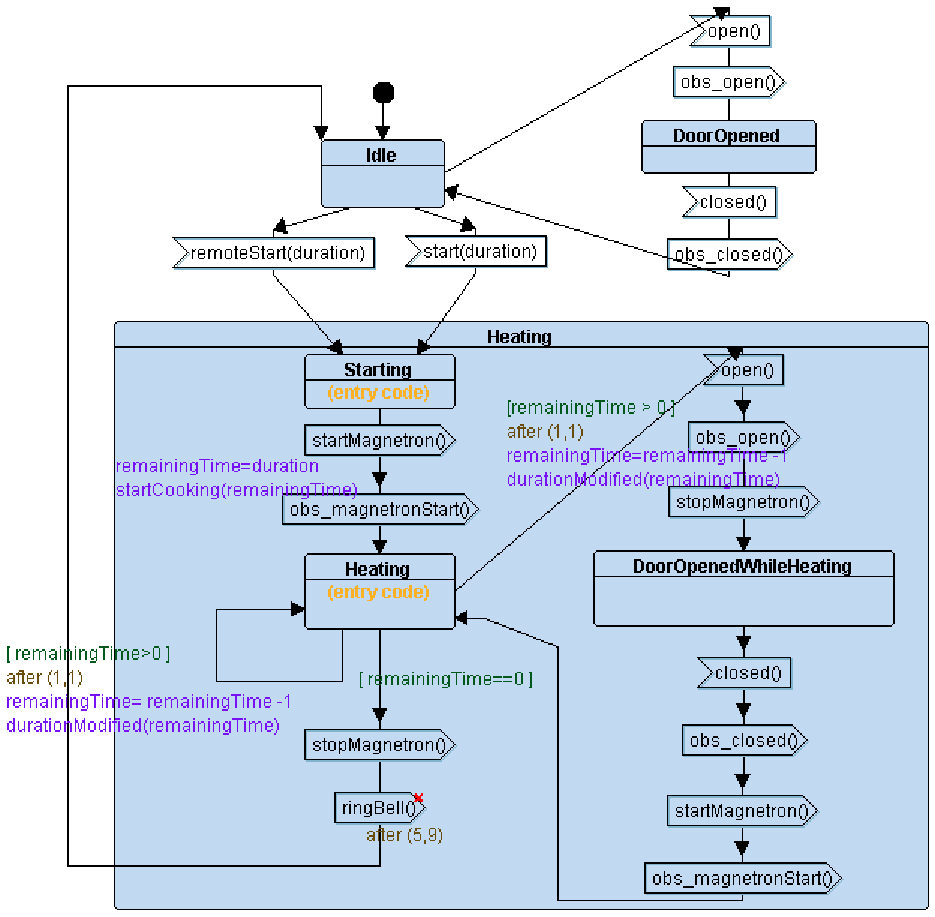
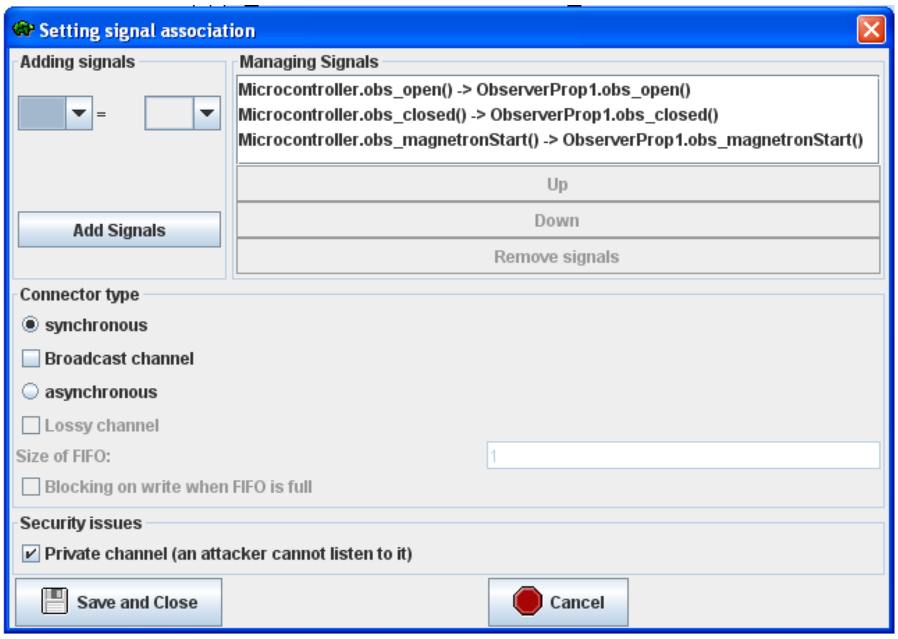


Figure 29: State-machine of the microwave microcontroller

The communication channel between the observer and the microcontroller is defined as synchronous and private, i.e. an attacker may not have access to this channel, in order not to compromise the assessment of security properties. A one-to-one correspondence is established between the signals in the micro-controller and in the observer, as shown in Figure 30.

[39] An alternative would have been to connect the observer directly to the door and to the magnetron. Considering that the communication between the door and the micro-controller, and the communication between the magnetron and the micro-controller are synchronous and lossless, both models are equivalent. If the communications are lossy, the alternative implementation of the observer will provide different results.



Setting signal association

Adding signals

Managing Signals

```
Microcontroller.obs_open() -> ObserverProp1.obs_open()
Microcontroller.obs_closed() -> ObserverProp1.obs_closed()
Microcontroller.obs_magnetronStart() -> ObserverProp1.obs_magnetronStart()
```

Connector type
- ● synchronous
- ☐ Broadcast channel
- ○ asynchronous
- ☐ Lossy channel

Size of FIFO: 1

- ☐ Blocking on write when FIFO is full

Security issues
- ☑ Private channel (an attacker cannot listen to it)

Figure 30: Signal association settings for the safety observer

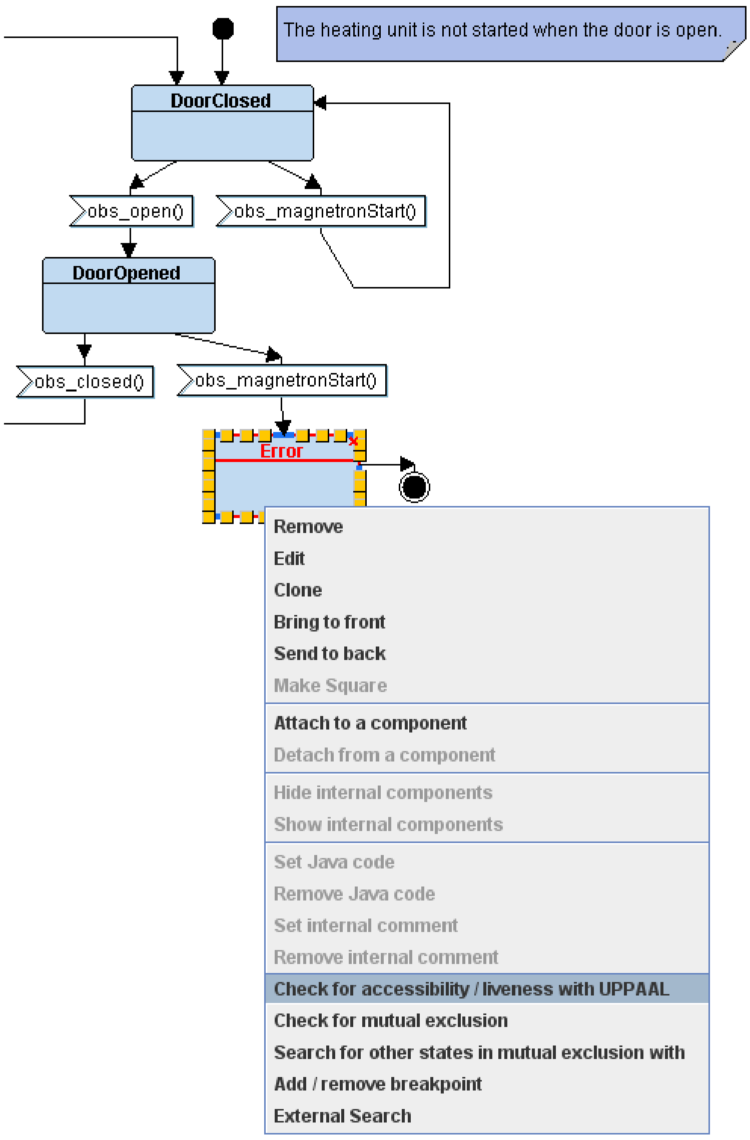


Figure 31: State-machine for the safety property n°1 observer

Now, to model the property with this observer, we need to reword the requirement as follows: “The state in which the observer arrives when it has observed that the door has been opened and the magnetron started whilst the door is still open, should not be reachable. This results in the state-machine shown in Figure 31 for the observer. A right-click on the “Error” state allows notifying UPPAAL that it should verify the reachability and / or liveliness of that state.

Assessment take-away

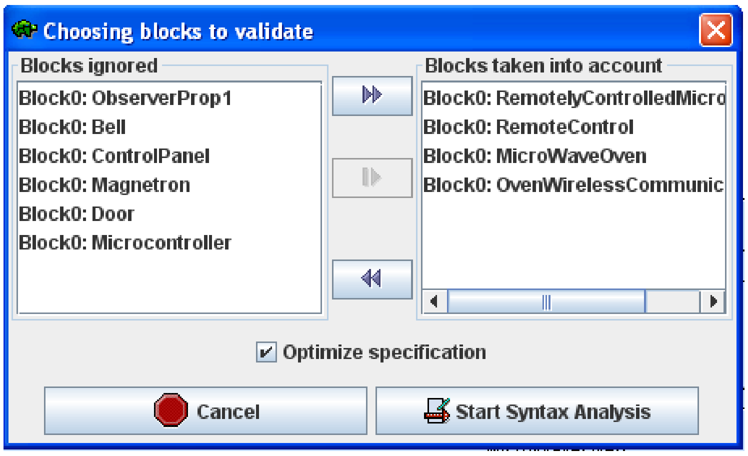
The AVATAR TEPE extensions to model safety properties are extremely interesting. However, TTool does not (yet) support the automated verification of properties expressed in TEPE. The AVATAR fall-back solution is based on the definition of *observers*. These observers perform the job, but they have significant detrimental effects on the readability of the system / software model, and thus cannot be considered as a long-term solution.

4.3.6 Formal verification

4.3.6.1 Prerequisites

Prior to performing a formal verification with a third-party tool, TTool must perform some syntax analysis of the AVATAR model (cf. Figure 32). This normally prevents the generation of incorrect code for ProVerif and UPPAAL.

Note: syntax analysis can also be useful to discover errors in the AVATAR model, at early stages of the design.


Choosing blocks to validate

Blocks ignored	Blocks taken into account
Block0: ObserverProp1	Block0: RemotelyControlledMicro
Block0: Bell	Block0: RemoteControl
Block0: ControlPanel	Block0: MicroWaveOven
Block0: Magnetron	Block0: OvenWirelessCommunic
Block0: Door	
Block0: Microcontroller	

[✓] Optimize specification

Cancel | Start Syntax Analysis

Figure 32: De-scoping the design model and syntax analysis

Another more subtle goal of the TTool syntax analysis is the de-scoping of the design model for the formal analyses. Indeed, the transformation of all AVATAR constructs is not (yet) supported. Some correct AVATAR constructs may trigger errors in ProVerif and / or UPPAAL. TTool offers a dialogue window in which it is possible to select the SysML blocks to take into account in the syntax analysis. This filtering is then (implicitly) used to limit the scope of the subsequent formal analyses.

The TTool syntax analysis also erases all back annotations on the AVATAR model.

Once the syntax verification is done, it is possible to launch the formal verifications using ProVerif and / or UPPAAL, as illustrated in Figure 33.

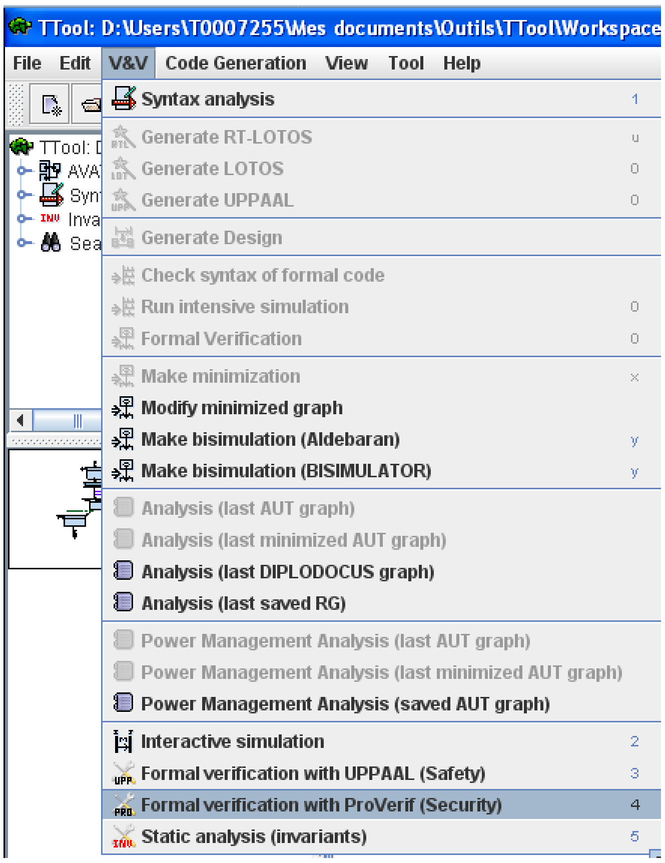

TTool: D:\Users\T0007255\Mes documents\Outils\TTool\Workspace
File Edit V&V Code Generation View Tool Help
Syntax analysis 1
Generate RT-LOTOS u
Generate LOTOS 0
Generate UPPAAL 0
Generate Design
Check syntax of formal code
Run intensive simulation 0
Formal Verification 0
Make minimization x
Modify minimized graph
Make bisimulation (Aldebaran) y
Make bisimulation (BISIMULATOR) y
Analysis (last AUT graph)
Analysis (last minimized AUT graph)
Analysis (last DIPLODOCUS graph)
Analysis (last saved RG)
Power Management Analysis (last AUT graph)
Power Management Analysis (last minimized AUT graph)
Power Management Analysis (saved AUT graph)
Interactive simulation 2
Formal verification with UPPAAL (Safety) 3
Formal verification with ProVerif (Security) 4
Static analysis (invariants) 5

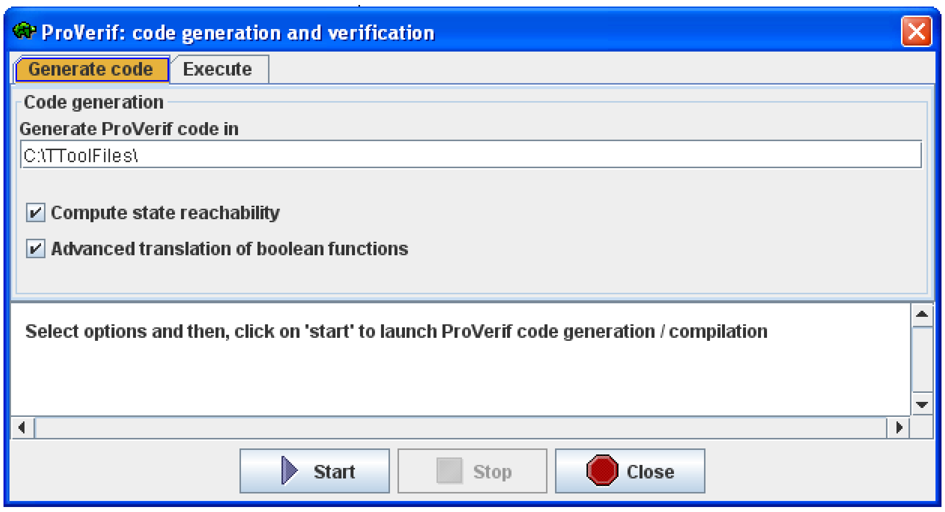
Figure 33: Menu to launch the formal verifications

4.3.6.2 Prerequisites to formal verifications using ProVerif

ProVerif can be used to verify:

- confidentiality properties, cf. §4.3.6.4;
- authenticity properties, cf. §4.3.6.5.

When the ProVerif option is selected in the menu (cf. Figure 33), a window opens to control the code generation, i.e. model transformation from AVATAR to ProVerif (cf. Figure 34).



Generate ProVerif code in
C:\TToolFiles\

☑ Compute state reachability
☑ Advanced translation of boolean functions

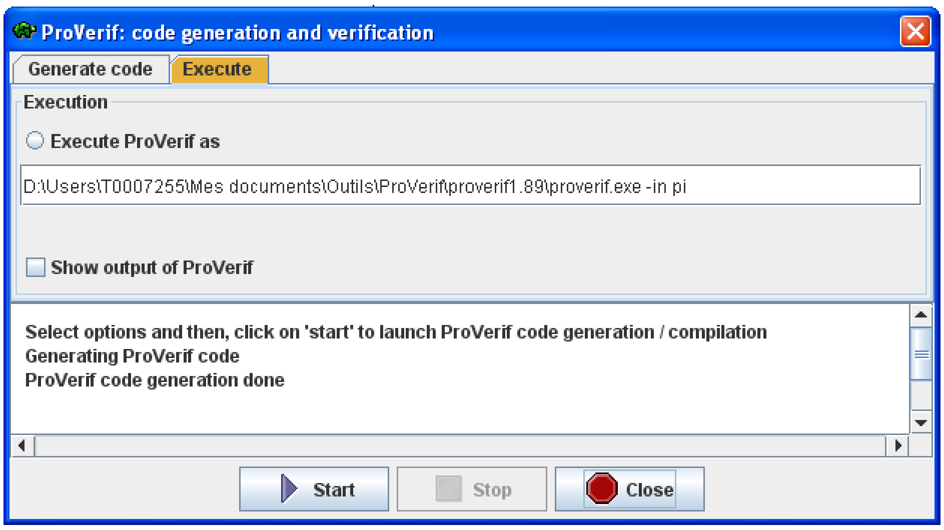
Select options and then, click on 'start' to launch ProVerif code generation / compilation

Figure 34: ProVerif's code generation dialogue window

This dialogue window recalls the directory in which the ProVerif code will be generated, in a file called "pvspec". This is useful to check the generated code[40], and potentially to launch ProVerif directly without using TTool.

The dialogue window also allows selecting an option to compute state reachability. This option is verbose, and should therefore be deactivated if one needs to analyse the traces generated by ProVerif.

After pressing "Start", TTool generates the code and a window opens to control the execution of the generated code by ProVerif, cf. Figure 35. Here, it is possible to ask to show all the traces produces by ProVerif, if one needs to understand the reasoning behind the results[41].



```
D:\Users\T0007255\Mes documents\Outils\ProVerif\proverif1.89\proverif.exe -in pi
```

```
Select options and then, click on 'start' to launch ProVerif code generation / compilation
Generating ProVerif code
ProVerif code generation done
```

Figure 35: ProVerif's code execution dialogue window

Pressing start will:

- launch the verification;
- provide the results textually in the window;
- back-annotate the AVATAR model with the results.

4.3.6.3 Prerequisites to formal verifications using UPPAAL

UPPAAL can be used to verify:

- the existence of deadlocks;
- the reachability of selected states;
- the liveness of selected states;
- the "leads to" property between two states[42];
- or any other custom property, as long as it can be expressed as a CTL formula.

When the UPPAAL option is selected in the menu (cf. Figure 33), a window opens to control the verification process, cf. Figure 36.

Assessment take-away

The AVATAR syntax verification was found to be very useful in debugging the model, even if it can still be improved to cover more cases.

[40] It is important to note that currently TTool does not check if the writing of the *pvspec* file is processed correctly. If the *pvspec* file is opened prior to code generation, code generation will fail, but no message will be produced. Thus, the AVATAR verification results will correspond to those of the opened file, not to the AVATAR model (if modifications have been made since the last generation).

[41] For simple cases only. A test with a complex protocol led to the generation of 1935 pages of traces…

[42] I.e. if the 1st state is accessed, then the 2nd one is eventually reached.

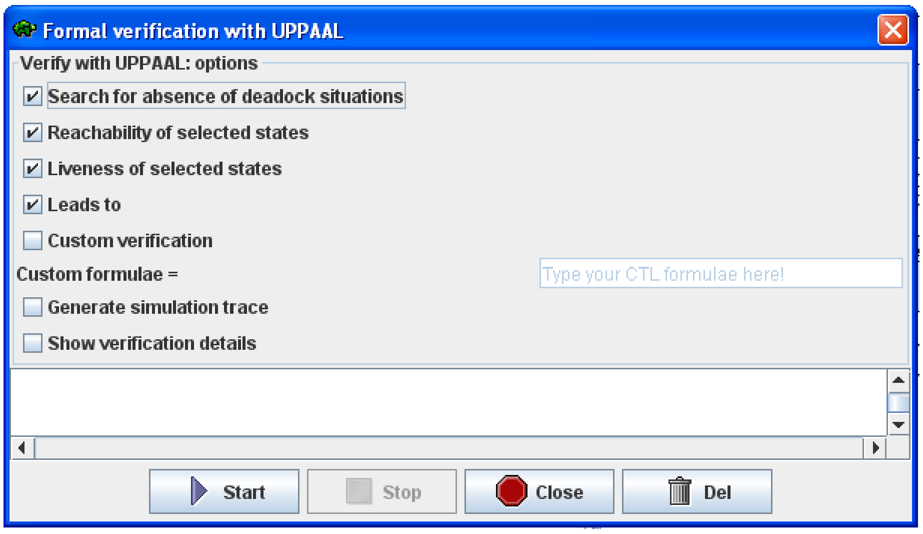

Formal verification with UPPAAL
Verify with UPPAAL: options
Search for absence of deadock situations
Reachability of selected states
Liveness of selected states
Leads to
Custom verification
Custom formulae =
Type your CTL formulae here!
Generate simulation trace
Show verification details
Start
Stop
Close
Del

Figure 36: UPPAAL's formal verification dialogue window

4.3.6.4 Confidentiality assessment

To assess the performance of AVATAR with respect to confidentiality, we have implemented three handshake protocols, as provided in (Dolev, et al., 1983). We start by briefly recalling these three protocols.

Let:

- A and B be communicating partners;
- $E_x(M)$ be the encryption of message M with the public key of X;
- XY be the concatenation of X and Y.

Protocol n°1:

$A \rightarrow B$: (A, E_B(M), B)

$B \rightarrow A$: (B, E_A(M), A)

Protocol n°2:

$A \rightarrow B$: (A, E_B(MA), B)

$B \rightarrow A$: (B, E_A(MB), A)

Protocol n°3:

$A \rightarrow B$: (A, $E_B(E_B(M)A)$, B)

$B \rightarrow A$: (B, $E_A(E_A(M)B)$, A)

Protocol n°2 is secure. Protocols n°1 and n°3 are both breakable, as shown below. Let Z be a saboteur.

MitM attack protocol n°1:

Z intercepts the message from A to B

$Z \rightarrow B$: (Z, E_B(M), B)

$B \rightarrow Z$: (B, E_Z(M), Z)

Z decodes E_Z(M), thus obtaining M

MitM attack protocol n°3:

Z intercepts the message from B to A, thus obtaining $E_A(E_A(M)B)$ here after noted E_A(N)

$Z \rightarrow A$: (Z, $E_A(E_A(N)Z)$, A)

$A \rightarrow Z$: (A, $E_Z(E_Z(N)A)$, Z)

Z decodes $E_Z(E_Z(N)A)$, thus obtaining N, and therefore E_A(M)

$Z \rightarrow A$: (Z, $E_A(E_A(M)Z)$, A)

$A \rightarrow Z$: (A, $E_Z(E_Z(M)A)$, Z)

Z decodes $E_Z(E_Z(M)A)$, thus obtaining M

The protocols were designed in AVATAR using two cryptoblocks A and B (cf. Figure 19), and implementing the protocols with state machines. Protocol n°2 is illustrated in Figure 37. The other protocols are similar and are not shown here. TTool / AVATAR provided the correct answers for all three protocols. The results are shown textually and graphically in Figure 38. This figure shows twice the A cryptoblock, once for the execution of protocol n°1 after which the secret n°1 is back annotated with a red lock, and once for the execution of the protocol n°2 after which the secret n°2 is back annotated with a green lock.

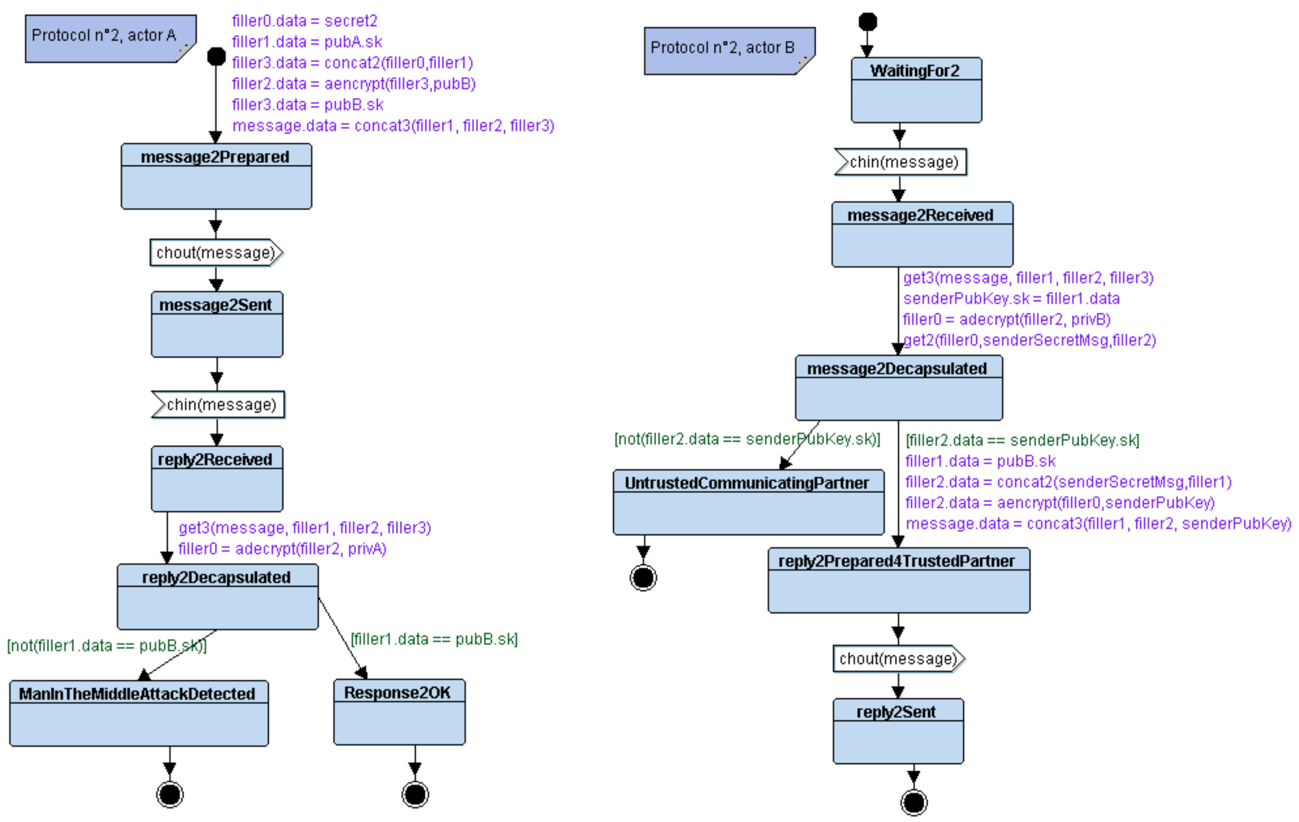
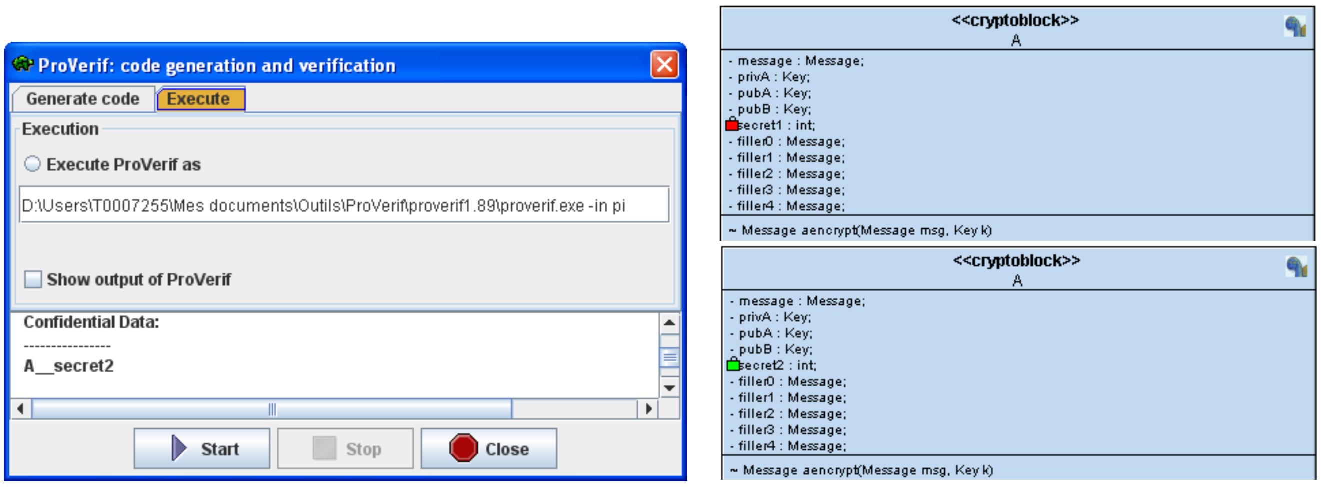

Protocol n°2, actor A
filler0.data = secret2
filler1.data = pubA.sk
filler3.data = concat2(filler0,filler1)
filler2.data = aencrypt(filler3,pubB)
filler3.data = pubB.sk
message.data = concat3(filler1, filler2, filler3)
message2Prepared
chout(message)
message2Sent
chin(message)
reply2Received
get3(message, filler1, filler2, filler3)
filler0 = adecrypt(filler2, privA)
reply2Decapsulated
[not(filler1.data == pubB.sk)]
[filler1.data == pubB.sk]
ManInTheMiddleAttackDetected
Response2OK
Protocol n°2, actor B
WaitingFor2
chin(message)
message2Received
get3(message, filler1, filler2, filler3)
senderPubKey.sk = filler1.data
filler0 = adecrypt(filler2, privB)
get2(filler0,senderSecretMsg,filler2)
message2Decapsulated
[not(filler2.data == senderPubKey.sk)]
UntrustedCommunicatingPartner
[filler2.data == senderPubKey.sk]
filler1.data = pubB.sk
filler2.data = concat2(senderSecretMsg,filler1)
filler2.data = aencrypt(filler0,senderPubKey)
message.data = concat3(filler1, filler2, senderPubKey)
reply2Prepared4TrustedPartner
chout(message)
reply2Sent

Figure 37: State-machines for handshake protocol n°2

During the implementation, one complexity that we faced is that the protocol narrations, as provided above, do not specify the checks that should be made by the participants during the execution of the protocol. If these checks are not made, then an attacker may exploit the vulnerability and therefore compromise the confidentiality of data. It is in particular the case for the test performed by actor B (cf. Figure 37), to abort the protocol in case of impersonation. If this test is not designed, the TTool / AVATAR signals the protocol as unsecure.



D:\Users\T0007255\Mes documents\Outils\ProVerif\proverif1.89\proverif.exe -in pi

```
Confidential Data:
----------------
A__secret2
```

```
<<cryptoblock>>
A
- message : Message;
- privA : Key;
- pubA : Key;
- pubB : Key;
🟥secret1 : int;
- filler0 : Message;
- filler1 : Message;
- filler2 : Message;
- filler3 : Message;
- filler4 : Message;
~ Message aencrypt(Message msg, Key k)
```

```
<<cryptoblock>>
A
- message : Message;
- privA : Key;
- pubA : Key;
- pubB : Key;
🟩secret2 : int;
- filler0 : Message;
- filler1 : Message;
- filler2 : Message;
- filler3 : Message;
- filler4 : Message;
~ Message aencrypt(Message msg, Key k)
```

Figure 38: Reports about confidentiality assessment

For fun:

- We leaked the private key of A as the 1st step of protocol 2; unsurprisingly, TTool / AVATAR correctly reported that the confidentiality of the secret n°2 was compromised.

- We chained the three protocols; much to our surprise, TTool / AVATAR reported that the confidentiality of the secret n°2 was now compromised, even though secret n°2 is only exchanged using the secure protocol n°2; looking at the ProVerif traces, it was seen that the attacker Z can retrieve $E_B(MA)$ during the exchange in protocol n°2, and then use for example the third protocol, to decipher secret n°2 as follows:
  Z → B: (Z, $E_B(E_B(MA)Z)$, B)
  B → Z: (B, $E_Z(E_Z(MA)B)$, Z)
  Z decodes $E_Z(E_Z(MA)B)$, thus obtaining M. Alternatively, the attacker can also use the first protocol to break secret n°2.

Assessment take-away

Overall, the evaluation results for confidentiality assessment are extremely positive. Three handshake protocols and a number of secure communications were assessed, and all results met our expectations.

4.3.6.5 Authenticity assessment

To assess the performance of AVATAR with respect to the authenticity of a message transmission, we checked the results returned by TTool on the authenticity pragmas defined in §4.3.5.1.

The initial results provided by TTool were erroneous results. Telecom Paris-Tech provided a patch after which it was shown that TTool did not differentiate weak authentication from strong authentication[43] in its result display panel. A second patch was provided by Telecom Paris-Tech to clearly differentiate both cases.

The output for the most complete Alice & Bob protocol with a signature and a nonce (cf. §4.3.5.1), is shown in Figure 39. These results are fully satisfactory.

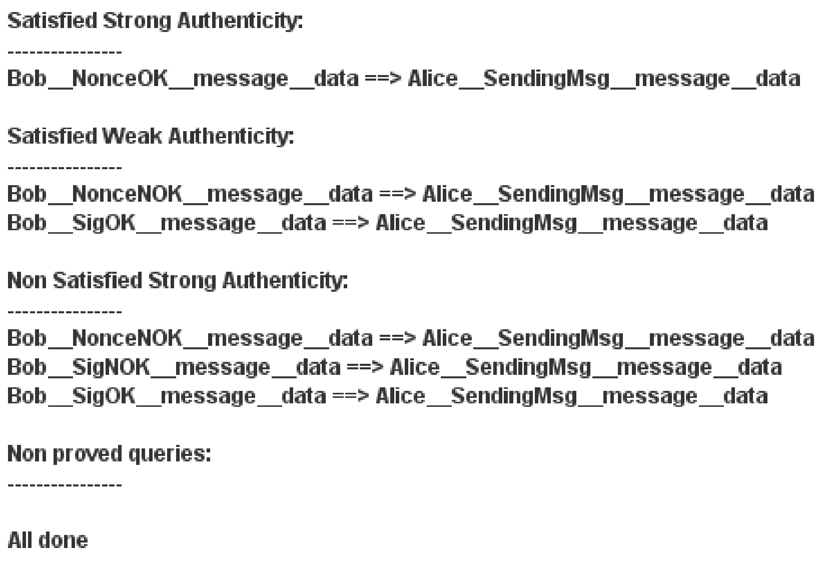
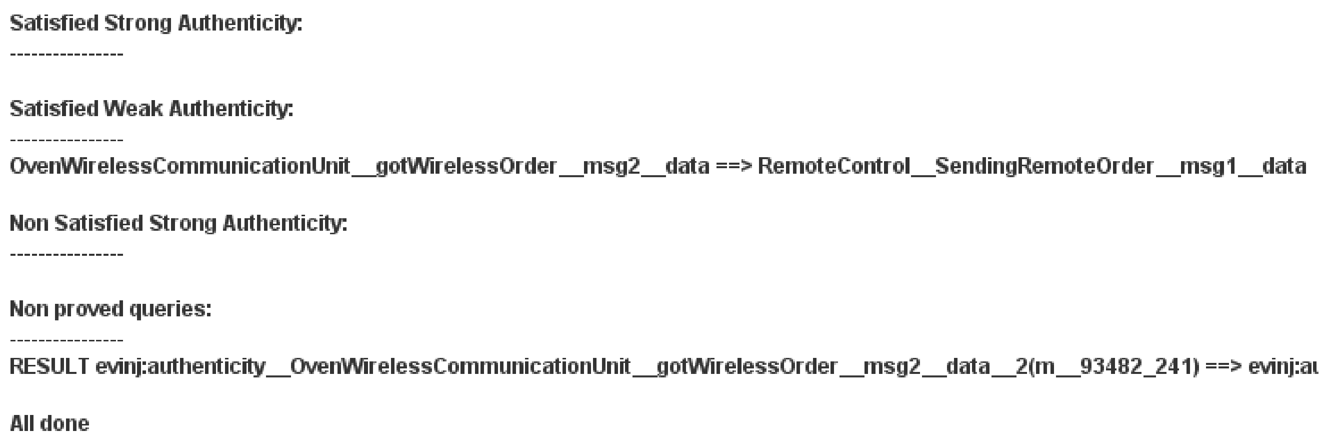

```
Satisfied Strong Authenticity:
----------------
Bob__NonceOK__message__data ==> Alice__SendingMsg__message__data

Satisfied Weak Authenticity:
----------------
Bob__NonceNOK__message__data ==> Alice__SendingMsg__message__data
Bob__SigOK__message__data ==> Alice__SendingMsg__message__data

Non Satisfied Strong Authenticity:
----------------
Bob__NonceNOK__message__data ==> Alice__SendingMsg__message__data
Bob__SigNOK__message__data ==> Alice__SendingMsg__message__data
Bob__SigOK__message__data ==> Alice__SendingMsg__message__data

Non proved queries:
----------------

All done
```

Figure 39: Reports about authenticity assessment on the Alice & Bob use protocol with nonce

The output for the microwave protocol with a symmetrical encryption, is shown in Figure 40. These results show a weak authentication and a non-proved query, which is a reasonable output considering the way ProVerif manages symmetrical encryption (cf. footnote in §4.3.5.1).

[43] See *Assessment Take-Away* below for definitions of weak and strong authenticity.



```
Satisfied Strong Authenticity:
----------------

Satisfied Weak Authenticity:
----------------
OvenWirelessCommunicationUnit__gotWirelessOrder__msg2__data ==> RemoteControl__SendingRemoteOrder__msg1__data

Non Satisfied Strong Authenticity:
----------------

Non proved queries:
----------------
RESULT evinj:authenticity__OvenWirelessCommunicationUnit__gotWirelessOrder__msg2__data__2(m__93482_241) ==> evinj:au

All done
```

Figure 40: Reports about authenticity assessment on the microwave protocol with a symmetrical encryption

Assessment take-away

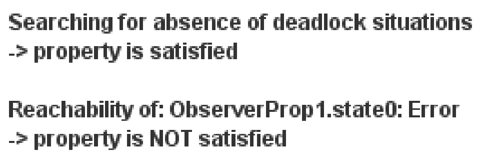
The formal verification of authenticity was time-consuming due to a number of bugs that required patches, but the results exceeded our expectations since AVATAR was shown to be able to differentiate weak authentication from strong authentication, with the following semantics:

- weak authentication assures that the message was forged by the communicating partner, i.e. message content and source authenticity,
- strong authentication assures, in addition, that the message was sent by the communicating partner, i.e. transmission authenticity, or in other words, the message has not being replayed.

Overall, the evaluation results for the formal verification of authenticity properties are therefore extremely positive.

4.3.6.6 Assessment of safety properties

To assess the performance of AVATAR with respect to safety properties, we checked the results returned by TTool on the safety property defined in §4.3.5.2. We checked for the reachability of the error state. As shown in Figure 41, the results were in line to our expectations.



```
Searching for absence of deadlock situations
-> property is satisfied

Reachability of: ObserverProp1.state0: Error
-> property is NOT satisfied
```

Figure 41: Results from the verification of the safety property

Assessment take-away

The formal verification of safety properties is straight forward and intuitive.

We however regret (again) that the formal verification is not based on a TEPE model, but on *observers* (cf. §4.3.5.2).

4.4 Conclusions

We conclude on a SWOT of the tool.

Strengths:

- Security constraints and properties, including the pre-sharing of secret information between a defined set of actors, are easy to formalise using AVATAR.
- Specific block stereotype, called <<*cryptoblock*>>, to help design secured communications. This stereotype provides the signature of the methods commonly used in designing secured communications.

- Methodology integrated inside the tool, with traceability between the methodological phases and the diagrams.
- Capability to capture informally stated assumptions, and trace them to model elements.
- Specific stereotypes, called <<*Safety Requirement*>> and <<*Security Requirement*>>, with ad-hoc tags, to manage SysML-based safety and security requirements.
- Two temporal operators that extend the SysML standard state-machines to support real-time system schedulability analysis.
- Stable tool.
- Syntax verification prior to transformation to ProVerif and UPPAAL.

Weaknesses[44]:

- Lack of documentation.
- ProVerif does not support loops in the state diagrams of SysML blocks[45]. ProVerif theoretically starts an infinite number of processes, so loops are not necessary considering the ProVerif semantics. However, loops may be necessary for safety proofs with UPPAAL. In this case, the combination of both safety and security proofs may become complex. For example, in the microwave oven example, blocks that have loops in their state-machines must be excluded when transforming the model the ProVerif format.
- Security constraints and properties are formally expressed as pragmas in UML notes, but UML notes do not have an identifier, so traceability between these elements and other model elements, e.g. requirements, is not possible.
- Communication is restricted to point-to-point communications: broadcast and multicast are not supported.
- All private channels are managed as a unique channel[46]: this may cause undesired interactions.
- Block properties can only be of type *Int*, *Bool*, *Timer*. This means that: (i) the application's control logic must be defined using only those simple types; the application's algorithms, requiring more complex data types, must be written in C code, as part of the body of methods.
- Refactoring of block, attribute and method names is not supported.
- Refactoring the names of diagrams is not well supported with respect to their traceability from the methodology diagram, since the traceability link is lost (without warning).
- ProVerif supports only one datum element at a time in a communication channel. The use of the '*concat*' and '*get*' functions of the TTool Cryptoblocks help in designing communication protocols with more complex flows, but this perturbs readability.
- The zoom capability on diagrams is poorly supported.
- The undo capability on diagrams is effective, but generates a change of diagram that is rather disturbing.
- With ProVerif, tests can only be performed on Booleans. Tests on other types must be handled using constant identifiers and pre-shared knowledge.

Opportunities:

- Safety properties can be formalised using the TEPE language, but the transformation of TEPE towards UPPAAL has not (yet) been implemented, thus forbidding any form of automated verification. There is a pressing need to transform TEPE properties into safety observers.
- A security requirement may reference an attack node in an attack tree; this ensures a basic traceability mechanism, but there is no automated check of the coverage of attacks. According to Ludovic Apvrille, the development of this feature is on the tool's evolution roadmap.
- There are some integration issues with UPPAAL: (i) if the UPPAAL licence is obsolete, TTool does not recognise it and considers that UPPAAL proved all properties to be true; (ii) some syntax errors[47] in the TTool code are not detected by the TTool syntax checker, but trigger a message stating that UPPAAL is incorrectly installed or configured. According to Ludovic Apvrille, the correction of these issues is on the tool's evolution roadmap.

[44] Please consider that weaknesses that are bugs in nature have been reported to the Ludovic Apvrille, and many have already been corrected.
[45] Télécom ParisTech has announced a patch to manage loops. Future deliveries should raise this major limitation.
[46] Télécom ParisTech has announced a patch separate private channels. Future tool versions should not suffer from such an issue.
[47] As for example, a semicolon at the end of an instruction on a transition.

- There are also some integration issues with ProVerif: currently, only the blocks involved in a security property verification must be selected before launching ProVerif. If more blocks are selected, a ProVerif compilation error may occur. According to Ludovic Apvrille, the correction of this issue is on the tool's evolution roadmap.
- Improve syntax checking, in particular with respect to authenticity properties.

Threats:

- The tool is hand-held essentially by one person, Ludovic Apvrille.

Annex A – Microwave Use Case Supplementary Data

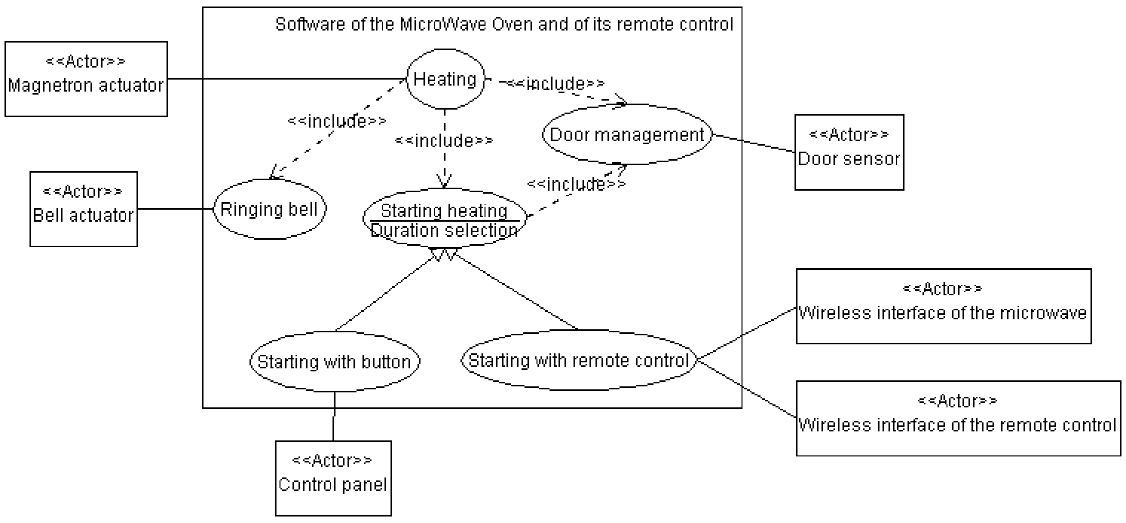

Software of the MicroWave Oven and of its remote control
<<Actor>> Magnetron actuator
<<Actor>> Bell actuator
<<Actor>> Door sensor
<<Actor>> Wireless interface of the microwave
<<Actor>> Wireless interface of the remote control
<<Actor>> Control panel
Heating
Door management
Ringing bell
Starting heating Duration selection
Starting with button
Starting with remote control
<<include>>

Figure 42: AVATAR Use Case Diagram for the microwave system

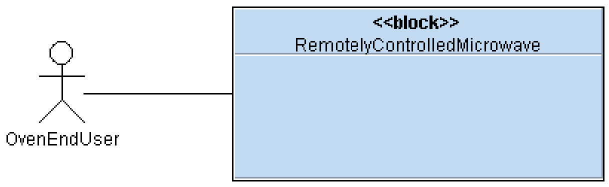


Figure 43: AVATAR Context Diagram for the microwave system

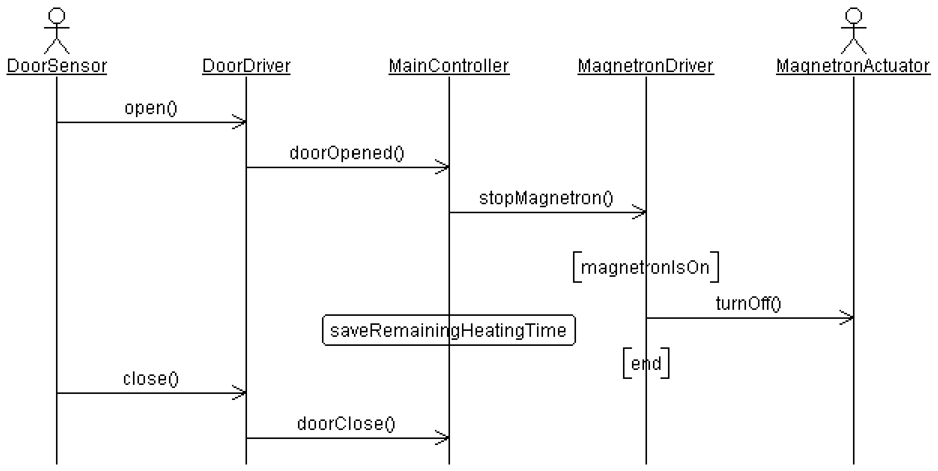


Figure 44: Example of AVATAR Sequence Diagram for the microwave system

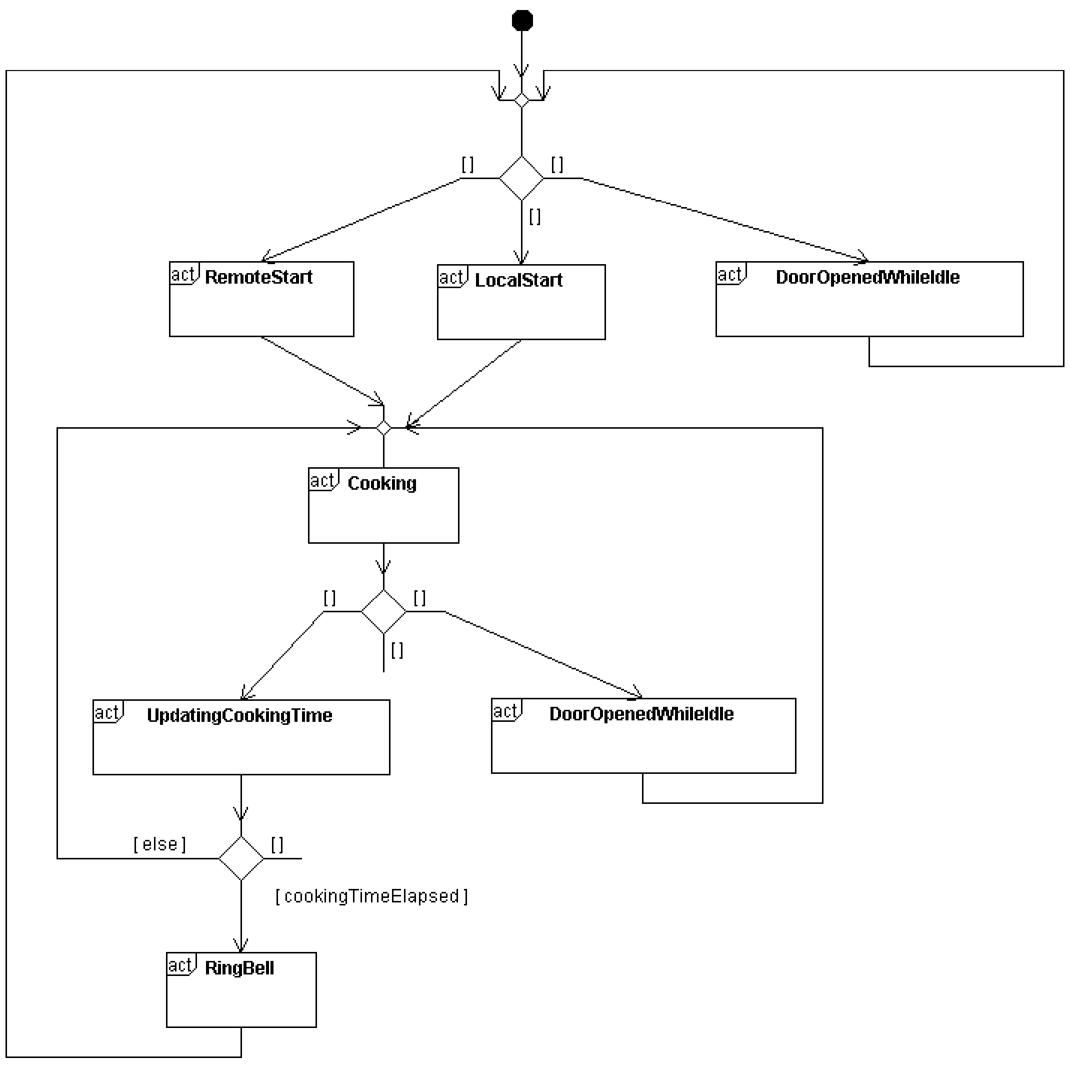


Figure 45: Example of AVATAR Activity Diagram for the microwave system

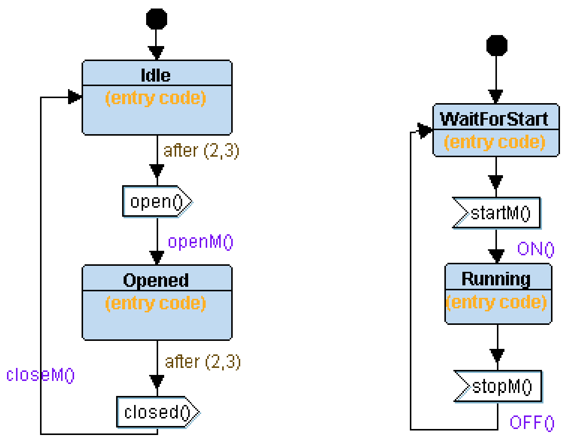


Figure 46: State-machines of the microwave’s door (left) and microwave’s magnetron (right)

Annex B – TEPE Supplementary Data

A specification in the TEPE language represents functional and non-functional properties in a formal way, using Parametric Diagrams (PDs). As opposed to informal SysML PDs, TEPE PDs are amenable to automated verification.

A small set of operators are leveraged to make up complex properties. TEPE operators manipulate three kinds of data:

- attributes, which are defined in blocks at system design level, or as new attributes, from existing ones.
- signals, which are defined in blocks at system design level, or as new signals, from existing ones, or which can be one of the two following additional signals: entry(state) and exit(state).
- properties, which are Boolean values resulting from SysML constraints, i.e. either equations, or temporal / logical constraint operators.

In TEPE, each property is expressed as a graph of signals, attributes, constraints and properties. This section presents rapidly the main TEPE concepts to support the reader in defining safety properties (cf. §4.3.5.1). For the TEPE methodology, please refer to §4.2.

Attribute-based operators

Two AVATAR operators allow for the definition of attributes in TEPE PDs:

- the <<attribute>> stereotype, when the attribute has already been defined in a block at system design level,
- the <<setting>> stereotype, for a new attribute.

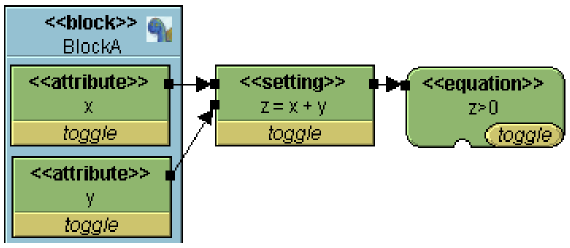


Figure 47: Attribute-based operators

The equation operator takes attributes as input, and it outputs a property.

Attribute operators output a signal indicating a value change (toggle).

An example of the three types of attribute-based operators is shown in Figure 47.

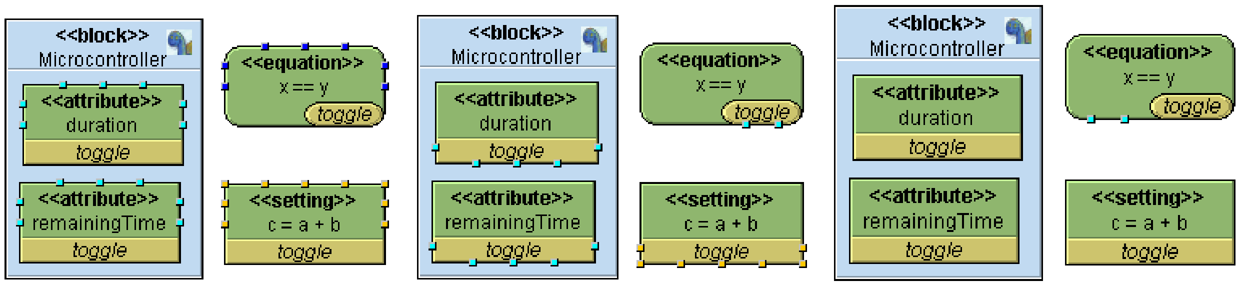


Figure 48: Attribute-based operators with their attribute ports (left), signal ports (middle) and property ports (right)

Since the attribute-based operators allow for different kinds of flows (i.e. attributes, signals and properties), they comprise different types of ports, which are highlighted by TTool, depending on the type of relation that needs to be drawn, cf. Figure 48. The ports are characterised by their position, which gives an indication of the type of flow, and by their colour, which gives an indication of the flow direction[48]. The colour code is as follows:

- cyan: output only;
- dark blue: input only;

[48]The ports on the toggle of <<setting>> should be "output only" (a patch will be provided).

- brownish: input / output.

Graphically, when a property flow is connected to a port of an operator, a white circle is displayed to recall that it is indeed a property flow (cf. Figure 52). This effectively increases readability.

In terms of ergonomics, TTool does not allow to select existing blocks and attributes from a list. The names need to be fully typed. This input modus operandi is error prone and should be revised.

Signal-based operators

Two AVATAR operators allow for the definition of signals in TEPE PDs:

- the <<signal>> stereotype, when the signal has already been defined in a block at system design level,
- the <<alias>> stereotype, for a merge of several distinct signals to one.

To evaluate signals, AVATAR offers three operators that translate the temporal behaviour of signals into properties:

- the logical constraint (LC) operator: this operator has for inputs two sets of signals (i.e. S_n, the normal input signals and S_f, the failure input signals), whose intersection is void, and an input property P_i (optional, considered to be true by default); the LC operator has for output one property P_o; once any signal S_{first} in S_n is encountered, the operator requires all signals $S_n \backslash \{s_{first}\}$ to be observed for P_o to be true; if none of the signals S_n is ever received, P_o is defined to be true; furthermore, the operator handles failure signals forcing P_o to be false in case they are notified between the first received signal of S_n and the last one; in addition to that, P_i is required to be true during all that period, otherwise P_o is set to false;
- the logical sequence (LS) operator: this operator works similarly to the Logical Constraint operator, apart from the fact that the order in which input signals are received is imposed;

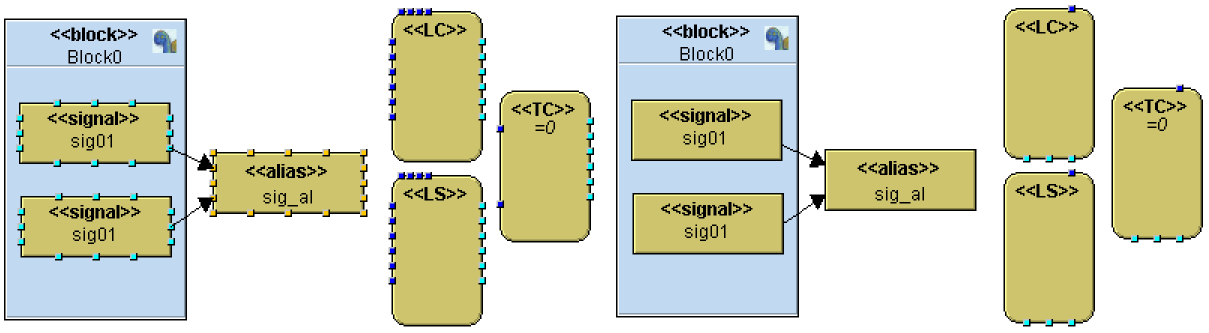


Figure 49: Signal-based operators with their signal ports (left), and property ports (right)

- the temporal constraint (TC) operator: this operator has for inputs two signals s_1, s_2 (the latter is optional), two time values t_{min}, t_{max} (either of the two is optional) and a property P_i (optional, considered to be true by default); the TC operator has for output one property P_o; depending on the provided arguments, P_o is defined to be true under the following conditions (cf. Figure 50):
  - s_2 has to occur at least t_{min}, at most t_{max} after s_1 and P_i must be true from the reception of s_1 to the reception of s_2;
  - s_2 has to occur at most t_{max} after s_1 and P_i must be true from the reception of s_1 to the reception of s_2;
  - s_2 has to be notified at least t_{min} after s_1 and P_i must be true from the reception of s_1 to the reception of s_2;
  - after reception of s_1, P_i must be true for at least t_{min} and at most t_{max};
  - after reception of s_1, P_i must be true for at most t_{max};
  - after reception of s_1, P_i must be true for at least t_{min}.

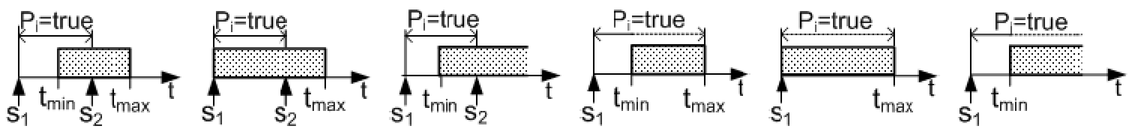


Figure 50: The different semantics of the TC operator

Graphically, the normal input signals (S_n) of the LC and LS operators need to be connected to the 6 ports on the left of the operator, whilst the failure input signals (S_f) need to be connected to the 4 ports on the top left of the operator. When a failure input signal is connected to a port on the top left of the operator, a small cross is displayed to recall that it is indeed a failure signal (cf. Figure 52). This effectively increases readability.

Property-based operators

Property operators comprise conjunction and disjunction functions for properties.

Property definition operators assign a name to a property, and specify its verification kind: (non-)reachability or (non-)liveness.

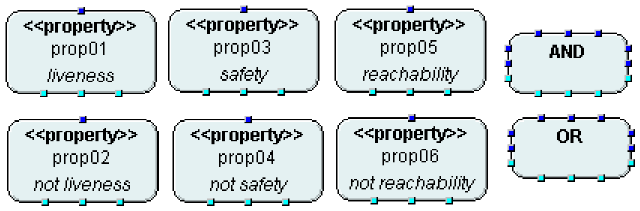


Figure 51: Property-based operators with their property ports

Annex C – Formalising Safety Properties using TEPE

To assess the capabilities of AVATAR in terms of safety property modelling, we provide the formalisation of two safety requirements (cf. §4.3.2) of our microwave running example. Each requirement is be covered by a property expressed using the TEPE language. Readers not familiar with the TEPE language should first read Annex B.

Formalisation example n°1

Our 1st example of safety requirement is "*The heating unit is not started when the door is open.*"

To ease the understanding of the property, please refer to:

- the state-machine corresponding to the *Microcontroller* (cf. Figure 29), which controls the *Magnetron*, and may be interrupted by the Door,
- the state-machines of the *Door* and the *Magnetron* (cf. Figure 46); in the state-machine of the *Door*, please note that the door is considered to be initially closed.

Now, the property corresponding to the formalisation of the requirement can be expressed as shown in Figure 52.

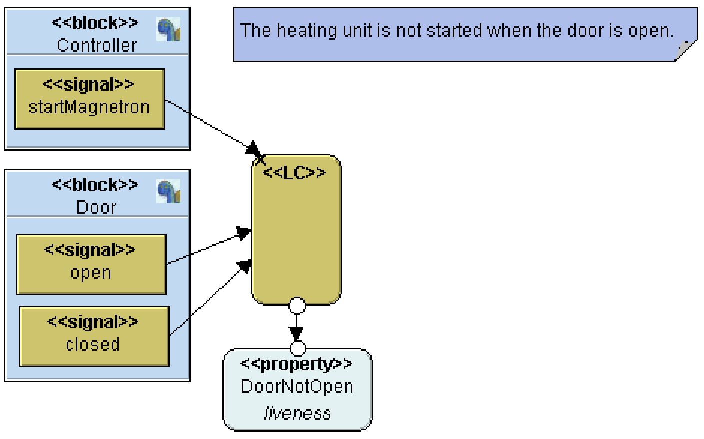


Figure 52: Property "Door not open"

With this expression:

- the property is true if the door is never opened;
- the property is true if the door is opened and closed before starting the magnetron,
- starting the magnetron when the door is opened will definitively make the property false, and thus the liveness of the property cannot be proved.

Formalisation example n°2

Our 2nd example of safety requirement is "*The bell rings only if the cooking time has expired.*" The property corresponding to the formalisation of the requirement can be expressed as shown in Figure 53.

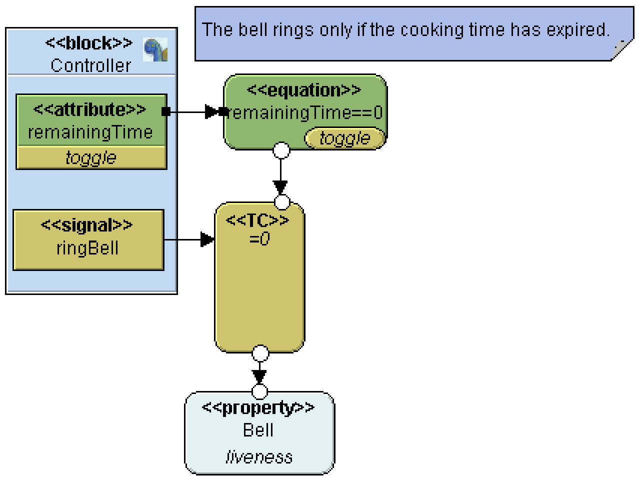


Figure 53: Property "Bell"

5 State of the art synthesis

This chapter provides an overall synthesis of the state of the art as presented in Part A of this deliverable.

Safety and security are two risk-driven activities that are traditionally tackled separately. It is thus possible to distinguish two communities, each working on their own standards, organising their own conferences, publishing in their own journals, and implementing on their own technical solutions. Since the 9/11 attacks on the Twin Towers in the Aeronautics domain and the discovery of the Stuxnet computer worm in the Industrial Control Systems domain in June 2010, it is more and more recognised worldwide that both engineering specialties cannot continue to ignore each other (cf. Figure 54).

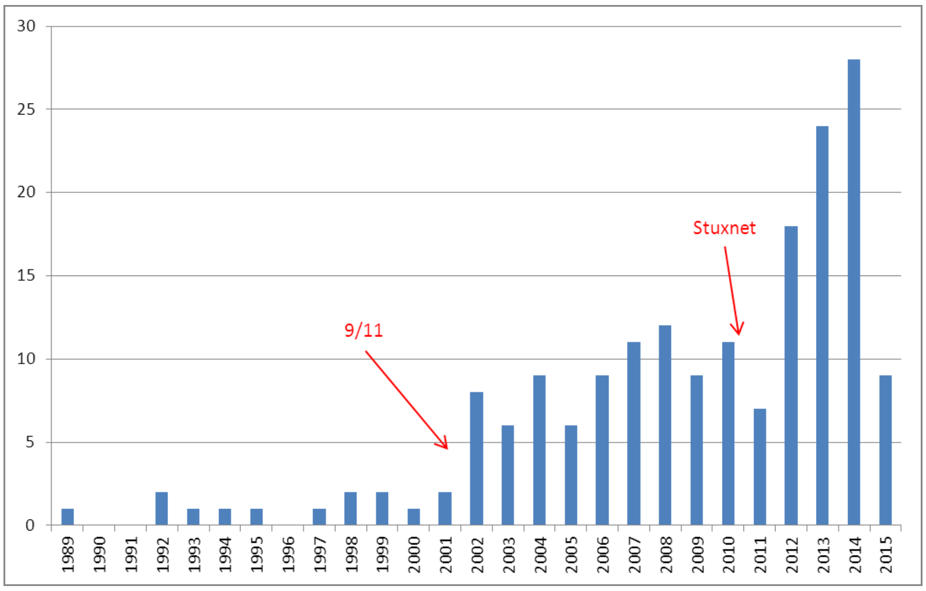


Figure 54: Publication dates of core referenced papers[49]

It is evident that there are major opportunities to share on onomastics, algorithms, (formal) methods and tools, in particular to reach higher levels of safety and security assurance at contained costs. Much work has already been done. This section provides a synthesis of the bibliography of research papers on safety and security engineering since the early 90's as reported in part A of this deliverable. The bibliography only covers papers that explicitly address both engineering specialties. Even papers dealing with dependability (Laprie, 1992) have been discarded if they do not explicitly mention safety and security.

The synthesis is organised in three groups. A first group (cf. §5.1) comprehends the papers that state the issues related to engineering safety and security separately, and assert that there is room for improvement, but do not explain how. The second group (cf. §5.2) comprehends the papers that propose to improve security engineering by adapting safety-related techniques, or vice-versa, in other words, safety and security cross-fertilisation. Here, one specialty is seen as more important than the other one, giving way to "security for safety" or vice-versa. The last set of papers (cf. §5.3) groups those that propose novel clean slate approaches for safety and security co-engineering, considering both specialties as peers.

Note: this synthesis was also been published as two papers at the SAFE'15 conference: (Paul, et al., 2015) and (Paul, 2015).

5.1 Houston, we have a problem!

A number of papers explicitly state the issues related to engineering safety without security or engineering safety and security separately, and assert that there is room for improvement, but they do not explain how (Pfitzmann, 2004), (Nordland, 2008), (Gerhold, 2011). Many of these papers are domain specific, e.g. (ICAO, 2005) in the Air Traffic Management (ATM) domain, (Deleuze, et al., 2008) with respect to industrial systems, (Bloomfield, et al., 2012) in the European Railway Traffic Management System (ERTMS) domain, (Koscher, et al., 2010), (Gebauer, 2014) and the National Highway Traffic Safety Administration (79 FR 60574, 2014) in the automotive domain, or (Vogt, 2014) on a Smart Grid case. Some paper pin-point the issue on very specific top-

[49] The 2015 figure is not as high as it should be, as our state of the art effort was relaxed beginning of 2015.

ics like safety vs. security metrics (Schwarz, 2014). Sometimes the issue statement is just given as a side-comment (Wiander, 2007). A striking fact is that the number of such papers does not seem to be diminishing as the years go by.

Beyond just expressing the issues, some papers also provide high-level recommendations on the manner to address them (Daniel, 2008), (Jalouneix, et al., 2009), (Carter, 2010), or on the directions to investigate, but they do not run that road themselves. Such recommendations advocate, e.g., an harmonisation[50] of safety and security requirements engineering processes (Eames, et al., 1999), (Smith, et al., 2003), the use of formal methods, and in particular the Liskov Substitutability Principle (LSP) when using an object-oriented paradigm and secure programming languages such as Ada / SPARK (Dewar, 2008), or on the contrary, extensive testing (Saglietti, 2008). Turning towards survivability engineering is also proposed (Goertzel, et al., 2009). Others are domain specific, such as (Schmittner, et al., 2014c) in the automotive domain.

In this very busy hive, (Taguchi, et al., 2015) steps back, proposing some high-level process patterns and case patterns corresponding to the main safety and security co-engineering approaches, but does not recommend the path to follow.

Running a bit against the current, a few papers, e.g. (Hansen, 2009), recall that even though safe systems were not designed to be secure, they often offer good properties against attacks, with a tendency to enter a fail-safe state rather than provoking accidents.

5.2 S4S: security for safety or safety for security?

This section is split in two parts: the first and most important one exposes papers that aim at improving safety engineering by considering security and / or privacy issues; the second exposes papers that aim at improving security engineering by integrating proven safety mechanisms.

(Piètre-Cambacedes, et al., 2013a) gives a comprehensive view of methods, models, tools and techniques that have been created in safety engineering and transposed to security engineering, or vice versa. The similarities and differences between the two domains are analysed. A careful screening of the literature (this paper contains 201 references) made it possible to identify cross-fertilizations in various fields such as architectural concepts (e.g. defence in depth, security or safety kernels), graphical formalisms (e.g. attack trees), structured risk analyses or fault tolerance and prevention techniques. (Kriaa, et al., 2015) also proposes a survey of approaches combining safety and security, but limiting the scope to industrial control systems.

5.2.1 Improving safety engineering

Safety engineering traditionally excludes malevolent behaviour; this is usually an implicit assumption, but it was sometimes explicitly stated, e.g. in the obsolete (IEC 61508-1, 1998) - (IEC 61508-7, 2000) standard series. Recent attacks in safety-critical domains, e.g. the 9/11 events in the Aviation domain, Stuxnet (Fallière, 2010) in the Industrial Control Systems (ICS) domain, have changed the game. The safety engineering community is addressing the issue by elaborating new focused techniques or wide-breath standards and guidelines, e.g. (S + IEC 61508, 2010), to seamlessly cope with IT security threats that can have an impact, direct or indirect, on safety. These techniques, standards and guidelines have major implications on the methods and tools used by industry to efficiently develop safety-critical systems; they usually render obsolete years of best practices, industrial quality baselines and require adequate training for the developers because the Security for Safety (S4S) approach is not a simple juxtaposition of safety and security processes.

It is possible to organise these focused techniques in two groups. The first group consists of established safety-related techniques that are enhanced to also cope with some security issues within a safety engineering process. The second group consists of security-related techniques that are adapted to enhance safety engineering.

In the first group, a focused safety-related technique that is often proposed for adaptation to cope with security issues is the HAZard and OPerability (HazOp) technique. A HazOp is a structured and systematic examination of a planned or existing process or operation in order to identify and evaluate problems that may represent risks to personnel or equipment, or prevent efficient operation. The adaptation usually comes down to defining new guide-words (Winther, et al., 2001), (Winther, 2004), (Srivatanakul, 2005), (Yang, et al., 2007), (Cusimano, et al., 2010), but can also be more comprehensive, as when it is used within the SeSa method (Grøtan, et al., 2007). Other adaptations relate to the *What-If* method (Yang, et al., 2007), Failure Modes and Effect Analysis, and Layer-Of-Protection Analysis (Hunter, 2009), or a combination of techniques (Brewer, 1993), (Srivatanakul, 2005), (Cusimano, et al., 2010). (Johnson, 2011) proposes the integration of security concerns into safety cases, combined with the use of Boolean Driven Markov Processes (BDMP) to avoid of the state explosion. (Netkachova, et al., 2015) also proposes security-informed safety cases. (Bezzateev, et al., 2013) and (Kornecki, et al., 2013a) suggest taking into account security hazards during the standard fault tree analyses and a similar approach is proposed in (Bieber, et al., 2014) with the extension of a safety-related Altarica model

[50] By opposition to the unification of the processes.

to allow for security analyses. (Contini, et al., 2006) proposes the use of non-coherent fault trees. (Gorbenko, et al., 2006) and (Babeshko, et al., 2008) present the Failure (Intrusion) Modes and Effects Analysis (F(I)MEA), whilst (Schmittner, et al., 2014b), and its shorter counterpart (Schmittner, et al., 2014a), presents the Failure Mode, Vulnerabilities and Effect Analysis (FMVEA) technique, which both extend the classical Failure Mode and Effect Analysis (FMEA) technique. (Roth, et al., 2013) proposes state/event fault trees (SEFTs) to allow for the modelling of vulnerabilities and an attacker model in complement to the traditional fault tree approach.

In the second group, (Johnson, 2004) claims that vulnerability assessment, traditionally used to improve security, can potentially provide new insights, a fresh and vivid perspective on safety hazards, and increased safety awareness. Likewise, (Sindre, 2007) and (Stålhane, et al., 2008) propose misuse cases to enhance safety engineering. Building upon this approach, (Raspotnig, et al., 2012a) proposes to adapt the security-related misuse sequence diagrams to support failure analysis. Pragmatically, (Mc Guire, 2011) recommends that the safety community looks how the open-source community has deployed multiple security methods (e.g. address space randomization) in order to simply apply those methods on their safety-critical systems.

Security specification is sometimes defined as the specification of what the system should not do, i.e. negative properties, e.g. non-interference in multi-level security. But negative properties are not an exclusivity of security. In safety, there are also numerous applications of negative properties, e.g. for the correct sequencing of operations[51] or the non-propagation of faults. Such security for safety approaches are proposed by (Rushby, 1989) and (Simpson, et al., 1998).

If the major part of the paper-contributions relates to the cross-fertilisation of safety and security techniques, there are also some novel and / or disruptive approaches, such as the introduction of the concept of *concern*, e.g. safety and security concerns, which are based on business goals and can drive the system requirements engineering process, thus filling the gap between the operational and system views (Sommerville, 2003). (Olive, et al., 2006) provides an overview of the Commercial Aircraft Information Security Concepts of Operation and Process Framework (ARINC 811, 2005), which attempts to educate and bridge the gaps among the safety and security disciplines by providing an understanding of airline operational constraints and an information security process, and serves as common framework for communication / coordination among stakeholders. Likewise, (Knorreck, et al., 2010), (Pedroza, et al., 2011), (De Saqui-Sannes, et al., 2011) and (Apvrille, et al., 2014) propose SysML-Sec, a SysML-based model-driven engineering environment that supports capturing and formally verifying security requirements, with particular attention being paid to their innocuousness with respect to safety requirements. Similarly, (Brunel, et al., 2014a) proposes an approach based on Alloy to formally model and assess a system architecture with respect to safety and security requirements; this approach was then extended (Brunel, et al., 2014b) to include Melody, a system engineering tool, and Safety Architect, a Failure Mode, Effects and Analysis (FMEA) tool, and packaged as a new framework called Coy (Brunel, et al., 2015).

Beyond the aforementioned focused techniques, there are various initiatives of the safety community which address the issue in a more comprehensive manner, in particular with respect to standards. In general, these initiatives are operational domain-specific, e.g. (SEISES, 2008), (Bieber, et al., 2012), (Paulitsch, et al., 2012) in the aeronautical, space and transport domains, (MODSafe, 2008) in the urban transport domain, (Bock, et al., 2012) in the railway automation domain, or (Goertzel, et al., 2011) for Navy weapon systems, even if sometimes the solutions may easily be extended to other safety and security-critical domains. We can distinguish two categories of initiatives. The first category defines new approaches that include security aspects whist maintaining compliance to existing standards. The second category defines new standards, or new versions of standards, that natively include security aspects.

Initiatives of the first category usually consist in analysing the gaps and overlaps between two (or more) existing standards in order to identify additional activities that need to be performed with respect to one standard used as baseline, in order to achieve dual compliance, e.g.:

- (Corneillie, et al., 1999) in relation to (ITSEC, 1991), (S + IEC 61508, 2010), (IEC 60880, 1986), (CENELEC EN 50128, 1997), (ETR 367, 1997) and (RTCA DO-178B, 1992) / (EUROCAE ED-12B, 1992);
- (Alves-Foss, et al., 2002), (Taylor, et al., 2002a) and (Taylor, et al., 2002b) in relation to (RTCA DO-178B, 1992) / (EUROCAE ED-12B, 1992) and the Common Criteria (ISO/IEC 15408-1, 2009);
- (Novak, et al., 2007) in relation to (S + IEC 61508, 2010) and the Common Criteria (ISO/IEC 15408-1, 2009);
- (Ridgway, 2007) in relation to (ISO/IEC 17799, 2005) and (BS EN 61508-1, 2002)
- (Derock, et al., 2010) in relation to (ISO/IEC 15026, 1998) and (ISO/IEC 27005, 2008);
- (Blanquart, et al., 2012) in relation to (ISO/IEC 27005, 2011), (SAE ARP 4754A, 2010) / (EUROCAE ED-79A, 2010), (EUROCAE ED-202, 2010), (ECSS-Q-ST-30C, 2009), (ECSS-Q-ST-40C, 2009), (ECSS-Q-ST-80C, 2009), (RTCA DO-178B, 1992) / (EUROCAE ED-12B, 1992), and Common Criteria (ISO/IEC 15408-1, 2009), but limited to the notions of safety levels and security levels, and with a focus on avionics;

[51] Seen under the security viewpoint as the avoidance of incorrect sequencing of operations.

- (Czerny, 2013) in the automotive domain, in relation to the (ISO 26262-1, 2011) - (ISO 26262-10, 2012) process framework.

Initiatives of the second category are essentially domain-specific. For example, the Airworthiness Security Process Specification (EUROCAE ED-202, 2010) / (RTCA DO-326, 2010) appeared as major contribution for *Security for Safety* engineering in the aeronautical domain. This standard was extensively discussed, e.g. in (Casals, et al., 2012), (Rowe, 2013) or (Joyce, et al., 2014) with respect to the methodology. It is noteworthy that the new edition of this standard (EUROCAE ED-202A, 2014) / (RTCA DO-326A, 2014) has significantly changed its recommendations in terms of co-engineering approach, moving from a *security sub-process* of the overall safety process, to a standalone *safety-informed security process*.

Likewise, in the railway domain, (CENELEC EN 20159, 2010) includes provisions for intentional attacks by means of messages to safety-related applications but it does not cover general IT security issues and in particular it does not cover IT security issues concerning the confidentiality of safety-related information, and the overloading of the transmission system.

Another significant standard is (S + IEC 61508, 2010) in the Electrical / Electronic / Programmable Electronic domain. The controversial nature of its security requirements has also been heavily discussed, e.g. in (Mc Guire, 2011) and (Schoitsch, 2014).

Our report is intrinsically about engineering. However, we felt it was important, before closing this section about security for safety, to give a little word about some embedded security mechanisms that are being proposed in a safety-critical domain, namely the automotive domain, even though it is difficult to relate these initiatives directly to co-engineering practices. Indeed, these initiatives, e.g. (Apvrille, et al., 2010b), (Groll, et al., 2010), (Stumpf, 2013), (Soja, 2014), do not explicitly appear as engineering techniques, but more explicitly as (essentially hardware) security solutions, whose purpose is increased safety. The above references are extremely light, and slightly out of scope herein, but we hope they give the reader a flavour of the whole swath of on-going *security for safety* solutions, which (hopefully[52]) result from not cited co-engineering studies.

5.2.2 Improving security engineering

Safety engineering is recognised as a more mature engineering speciality than security engineering. Thus, multiple authors propose to adapt safety engineering techniques to the security domain. Adaptations cover a wide range of techniques, from the socio-technical domain, e.g. (Brostoff, et al., 2001), (Fruth, et al., 2014), the revision of methodology, e.g. with the introduction of security-critical levels in (Gutgarts, et al., 2010), up to purely technical approaches, as described below.

The most recurrent safety-related technique that has been adapted for security engineering purposes is the HAZard and OPerability (HazOp) technique (Lynch, 2002). Adaptation is typically realised by defining new guide-words (Foster, 2002), (Lano, et al., 2002), (Srivatanakul, et al., 2004). Lessons learnt seem systematically positive, even though somehow contradictory, e.g. (Daruwala, et al., 2009) vs. (Foster, 2002). Other common techniques include the deviation analysis approach, as used in fault tree analysis (Foster, 2002), (Helmer, et al., 2002), (Brooke, et al., 2003), (Murdoch, et al., 2006) to support the investigation of potential vulnerabilities. Because no system-level methodology currently exists that can quantify the amount of security provided by a particular system-level approach, (Nicol, et al., 2004) proposes to adapt concepts and methodologies, normally used for the quantitative evaluation of system dependability and frequently based on stochastic modelling, to security evaluation. An alternative approach to the quantitative assessment of the effect of security breaches on a computer system, based on fault trees, is proposed in (Rushdi, et al., 2004) / (Rushdi, et al., 2005).

Beyond specific techniques, some papers have a more comprehensive approach by adapting the overall good practices and lessons learnt of safety engineering to security engineering (Axelrod, 2011). In the same spirit, (Young, et al., 2014) also addresses the security engineering improvement challenge by proposing a significant paradigm shift for security experts: the use of a systems-theoretic approach, shifting the majority of security analysis away from guarding against attacks (tactics) and more towards design of the broader socio-technical system (strategy).

5.3 Towards safety and security co-engineering

This section analyses the papers that propose novel approaches for safety and security co-engineering, considering both specialties as peers.

Amongst the first communities to address the relations between safety and security was the formal methods community, with the challenge of formalising the concepts, the mechanisms employed to safeguard them, and their interplay (Rushby, 1989), (Burns, et al., 1992), (Rushby, 1994), (Stavridou, et al., 1998). A key outcome of these studies is the formalisation of non-interference specifications and invariants (Ramirez, et al., 2014). This early work is closely related to the currently active research on the Multiple Independent Levels of Security

[52] It is unclear how far these security controls have been established as innocuous with respect to functional safety.

(MILS) architecture (Boettcher, et al., 2008), (ISO 25119-2, 2010), (EURO-MILS EC FP7 Project, 2012), (Müller, et al., 2012b), (Müller, et al., 2014), and its distributed version (D-MILS, 2007), (Cimatti, et al., 2015), for which successful stories are starting to appear (Müller, et al., 2012). Initially developed for security, this architecture also displays good safety properties, bringing more and more the term "MILS" to be used as a noun, rather than as an acronym. The proponents of this architecture claim that it is a good starting point to look for synergies & divergences for safety and security (Tverdyshev, 2014). Beyond MILS, recent fundamental advances in the formal methods community, including advances in satisfiability (SAT) and satisfiability modulo theories (SMT) solvers, separation logic, theorem provers, model checkers, domain-specific languages and code synthesis engines suggest that developing a high-assurance software workbench based on a combination of formal methods is now possible, as shown in (Fisher, 2013), (DARPA I2O HACMS, 2014), including a runtime assurance architecture with machine learning mechanisms (Tiwari, et al., 2014). Less industrial but still quite comprehensive, (Delange, 2010) proposes a framework based on: (a) the Architecture Analysis and Design Language (AADL) as unique representation language; (b) automated validation of the specifications; (c) code generation for execution on an open-source partitioned operating system (POK Community, 2011); (d) automated certification, which verifies that specification requirements are met in the implementation by analysing the system during its execution and also evaluates its compliance against certification standards. Finally, (Sun, et al., 2009) focuses only on requirements, proposing a formal framework that assists designers in detecting conflicts between safety and security requirements. A comprehensive review of Formal Methods for Safe and Secure Computers Systems is given in (Garavel, et al., 2013).

Some studies are less formal, but have the similar goals of better understanding the relations between safety and security (Pan, et al., 2007a), (Piètre-Cambacédès, et al., 2009), and establishing a common information model for safety and security (Avizienis, et al., 2004), (Jonsson, 2006), (Stoneburner, 2006), (Firesmith, 2010) and (Mattila, 2013). As early as 1992, (Jonsson, et al., 1992) was asking: "*Should we […] look for unification of terminology, or is it justifiable to maintain separate terminologies for each discipline?*" The important characterisation differences between safety and security concepts, e.g. as expressed in (Burns, et al., 1992) and in (Piètre-Cambacédès, et al., 2009), tend to show that it is extremely difficult to provide a comprehensive picture. (Firesmith, 2003) rightly points out that the information models of safety, security, and survivability engineering are remarkably similar in both content and topology, and therefore safety, security, and survivability requirements can be elicited and analysed in terms of a risk-oriented, asset-based approach that takes into account the associated hazards and threats from which these assets must be protected. However, this comprehensive approach brings the author to reconsider some rather standard definitions, to ensure overall consistency[53]. For example, security is decomposed into the following sub-factors: access control, attack/harm detection, availability protection, integrity, non-repudiation, physical protection, privacy, prosecution, recovery, security auditing, and system adaptation. This differs significantly from the traditional focus on the confidentiality, integrity and availability sub-factors, or even the extended list of security goal sub-factors proposed in Octave (Cherdantseva, et al., 2013). In particular, availability protection refers to the degree to which various types of Denial of Service (DoS) attacks are prevented, whereas availability in the safety-tradition deals with the operational availability of the system when it is not under attack. A solution might lie in the partition proposed by (Chapon, et al., 2012) or (Sadvandi, et al., 2012), i.e. the use of formal methods to address known and controlled risks (e.g. internal system faults, script kiddies), and in-depth defence, to address unknown or uncontrolled risks (e.g. causes external to the system, 0-day threats). Noteworthy as well, but focused on the automotive domain, (Schmittner, et al., 2015b) identifies three requirements to select a candidate security standard to complement a safety standard: (i) there should be an overlap in required work products for safety and security argumentation; (ii) assurance levels between safety and security should be translatable; (iii) approaches and concepts from one standard should be mirrored by the other standard.

Unifying focused engineering techniques used in safety and security is often recommended, as in (Lano, et al., 2002) around the implementation of HAZard and Operability (HAZOP) studies and Fault Tree Analyses (FTA) in the Unified Modelling Language (UML), or as in (Fovino, et al., 2009), (Förster, et al., 2010) with the integration of attack trees within fault trees. In this context, (Steiner, et al., 2013) solves the problem of the missing security events probabilities by the use of a hybrid rating scheme. (Piètre-Cambacédès, et al., 2010) proposes a similar unification with Boolean logic Driven Markov Processes (BDMP). Likewise, (Reichenbach, et al., 2012) proposes an approach for combining safety analysis with security analysis by considering the Safety Integrity Levels (SIL) of (S + IEC 61508, 2010) as an extension of the Threat Vulnerability and Risk Assessment (TVRA) method. In the automotive domain, (Macher, et al., 2015a) / (Macher, et al., 2015b) proposes the SAHARA (Security-Aware Hazard Analysis and Risk Assessment) approach, a combination of the well-known safety-centric HARA (Hazard Analysis and Risk Assessment) method and the security-centric Microsoft STRIDE method. An alternative approach for safety and security co-engineering in the automotive domain is presented in (Ward, et al., 2013).

Unification is usually proposed based on a selected set of safety-related and security-related techniques. Few papers however propose an exhaustive review of techniques to justify why specific attention is given this or that

[53] Likewise in (Jonsson, et al., 1992).

technique. (Raspotnig, et al., 2013a) provides an extensive (50 pages) review of risk identification techniques for safety and security requirements. The added-value of the article is that it proposes an assessment framework. All techniques are assessed against the selected criteria to obtain knowledge on strengths and weaknesses of the different techniques in both the safety and security fields, and suggestions are provided to mutually enhance their efficiency.

Even if a unification or harmonisation of the safety and security engineering approaches is commonly proposed, disruptive approaches are also proposed. (Zafar, et al., 2005) proposes the use of the Genetic Software Engineering (GSE) method to deal with the formal writing and verification of safety and security requirements. (Sallhammar, et al., 2006) presents a stochastic model for integrated security and dependability assessment using stochastic game theory which allows for the computation of the expected attacker behaviour. (Aven, 2009) claims that it is necessary to use a risk-informed approach where calculated probabilities and expected values are enriched with the uncertainties of the underlying phenomena and processes. (Monakova, et al., 2012) extends classical Business Process Modelling Notation (BPMN) to cope with safety and security requirements. (Subramanian, et al., 2013) and (Subramanian, et al., 2014) proposes the Non-Functional Requirements (NFR) technique that allows simultaneous evaluation of both safety and security at the architectural level, using respectively qualitative and quantitative reasoning to evaluate whether the properties have been achieved. (Kornecki, et al., 2013b) compares the traditional Non-Functional Requirement (NFR) approach with a Bayesian Belief Network (BBN) approach, which can be used when the factors related to the safety and security of cyber-physical systems are assumed to be randomly distributed. (Vouk, 2013) asserts that engineers appear to avoid and eliminate vulnerabilities more by luck (aleatoric process) than through knowledge driven (epistemic) methods; this opens some interesting models, e.g. for vulnerability detection or estimating the number of residual security faults. (Pieters, et al., 2014) proposes to quantify frequencies of targeted attacks in order to integrate security risk assessment methods in existing safety risk management practices, and support countermeasures investment decisions. (Schneider, 2014) proposes a contract-based approach called ConSerts to address the challenges of openness and runtime adaptation which are common the safety and security critical systems. (Kriaa, et al., 2015b) proposes S-cube, a framework to automatically generate the different attack and failure scenarios a system is exposed to, based on the system description.

Beyond the aforementioned focused techniques, there are various proposals for an overall unification. (MAFTIA, 2000) proposes a framework that ensures the dependability of distributed internet applications in the face of a wide class of faults and attacks. (Sørby, 2003) and (Horn, 2005) propose a development process for security-safety critical systems, which is based on the safety lifecycle defined in (IEC 61508-1, 1998) and the CORAS integrated risk management and system development process (Braber, et al., 2003). (Hessami, 2004) proposes a new paradigm for holistic systems assurance. (Altran Praxis, 2006), (Cockram, et al., 2007) and (Jackson, et al., 2008) present SafSec that helps achieve certifications with the minimum of duplicated work. (Ibrahim, et al., 2004) unifies the Capability Maturity Model Integration (CMMI) and the FAA integrated Capability Maturity Model (iCMM), whilst (Firesmith, 2010) relates safety and security engineering to survivability engineering. Likewise, (Raspotnig, et al., 2012b) proposes a unified process for the elicitation and analysis of safety and security requirements, called the Combined Harm Assessment for Safety and Security of Information Systems (CHASSIS) method, that comprehends three modelling techniques (Raspotnig, et al., 2012a) and a Harm Assessment Process (Raspotnig, 2014); the latter was extended by (Katta, et al., 2013a) with a security requirements traceability capability built upon the Safety Traceability Approach (Katta, et al., 2013b). (Pedroza, et al., 2011) proposes a unified tooled-up framework based on SysML for the specification of embedded systems, integrated with ProVerif and UPPAAL respectively for the verification of security and safety properties. (Sadvandi, et al., 2012) proposes a safety and security integrated paradigm in which formal risk assessment frameworks may be used to cover both safety and security known threats, and defence in-depth may help to mitigate both safety and security hardly-predictable risks. Based on the lessons learnt from the Stuxnet malware, (Aoyama, et al., 2013) proposes a novel framework tackling plant safety and security from a more comprehensive point of view. (Axelrod, 2012), (Axelrod, 2013b) and (Axelrod, 2013c) propose an approach to model cyber-physical systems and measure the risks to which they are exposed in order to better minimise total risk. (Woskowski, 2014) proposes to extend beyond device boundaries the (ISO 14971, 2007) risk-based approach related to the integration and interaction of medical devices. And there are many more approaches, such as (Schoitsch, 2005), (Line, et al., 2006), (Aven, 2007), (Aven, 2011), (Förster, et al., 2010), (Aoyama, et al., 2013).

(SeSaMo, 2012) proposes to develop a component-oriented design methodology based upon model-driven technology, jointly addressing safety and security aspects and their interrelation for networked embedded systems in multiple domains; in (SeSaMo D2.1, 2013), eighteen basic building blocks (BBs) for safety and security modelling are proposed, whilst (SeSaMo D3.1, 2013) provides a specification of safety and security analysis and assessment techniques. (SeSaMo D4.1, 2014) presents the safety and security integrated design and evaluation methodology. SeSaMo stands slightly apart from the other approaches in that it proposes the definition of *interaction points* between separate safety and security processes, rather than a unified process, but with shared and unique work-products (Mazzini, et al., 2014). Highlights are presented in (Favaro, et al., 2014), including security-informed safety cases. However, this genericity has it limits: specific domains have specific constraints and specific capabilities which can dramatically impact the design approaches, as shown in (Banerjee, et al., 2012) in the case of cyber-physical systems (CPSs).

It is difficult to assess which approach will emerge as we believe that the ultimate approach to co-engineering has not yet been found. (Kriaa, et al., 2013) supports this statement by comparing CHASSIS and BDMP, and concluding that these two approaches complement each other, thereby showing that neither of these two already integrated approaches is the ultimate solution. Likewise, (Schmittner, et al., 2015a) presents a case study combining FMVEA and CHASSIS, concluding that one weakness of CHASSIS is that, while safety and security are analysed with the same methodology, the two assessments are unacceptably done separately; moreover both methods do not explicitly address how to conduct safety and security analysis in a continuous manner.

Unification initiatives can also be found in standards, with e.g. (ISO 31000, 2009) and (IEC 31010, 2009) for risk assessments or (ISO/IEC 15026-2, 2011) and (OMG SACM, 2013) for assurances cases. In domains in which compliance to standards is of utmost importance, generic co-engineering approaches and technical solutions as presented above are rarely helpful, especially when one starts searching for the devil in the details. In this context, (Åkerberg, 2011) proposes an end-to-end safe and secure communication solution for standard-compliant heterogeneous automation networks, whilst (Braband, 2014a) and (Braband, 2014b) propose an IT security framework compliant with the safety standards in the railway automation domain.

Of course, when both safety and security concerns are addressed for a giving system, striking the proper balance between these two, sometimes contradictory, sets of requirements may be a challenge. (Nielson, et al., 2013) proposes an extension of the Quality Calculus to check the extent to which safety and security goals have been met. (Labreuche, et al., 2005) proposes a generic framework using multi-criteria decision aiding (MCDA) techniques based on the two-additive Choquet integral to help decision makers select the best option amongst several alternatives; the main author is now working, as part of the (MERgE, 2012) project, on a specific adaptation of this framework to support design decisions in the context of multi-concerns system architecting (not yet published).

All the above references relate to design-time engineering activities. Let us close the show by citing (Pan, et al., 2007b), a borderline paper with respect to this state of the art, which focuses on engineering activities to keep a system safe & secure during system operation and system maintenance.

5.4 Conclusion

As can be seen from the above, the academic and standardisation communities are active as never before on the subject of safety and security co-engineering. The subject usually raises much interest, even if there are from time to time some signs of disillusion. E.g., in 2005, Erwin Schoitsch published *Design for safety and security of complex embedded systems: a unified approach* (Schoitsch, 2005). Close to ten years later, the same author asks: *Safety and security – what about a joint process?* (Schoitsch, 2014). In the mass of publications, it is difficult to find technical safety and security development roadmaps –roadmaps we found, such as (Johnson, 2012) or (Luiijf, et al., 2015) target governmental policymakers or senior executives. It is equally difficult to predict the future, from a technical standpoint, based on the many directions that research is investigating. However it is possible to state a couple of facts, and we ventured to formulate a couple of trends.

The first fact is that safety and security co-engineering seems to be primarily a concern of the safety engineering community. Indeed, the increasing number of cyber-attacks in the world tends to show that safety-critical systems, and in particular the rising number of cyber-physical systems, which are particularly exposed by nature, may not be as safe as they claim, if they are not also secure. The multiplication of security-related workshops in conjunction to safety-related conferences, and the multiplication of safety standards updates that include security concerns both provide significant testimonies of this growing interest for safety and security co-engineering by the safety community. There is no similar earthquake within the security community: security experts seem to be interested in safety studies in two cases: (i) to assess if safety-critical systems are more vulnerable when they switch into fail-safe modes; (ii) to re-use safety techniques when availability and integrity are the primary concerns of the security engineering work, by opposition to confidentiality.

The second major fact is that the security regulation, with the exception of privacy regulation (Directive 95/46/EC, 1995), (EU COM(2012) 11 final, 2012), is somehow lagging behind industrial initiatives to produce security standards for software-intensive systems. Indeed, security is a National sovereignty prerogative, whilst safety regulation has been transferred to transnational organisations (e.g. European Commission, ICAO) since decades. Depending on the domains, National regulation may be seen as too weak or on the contrary an effective means to affect worldwide businesses. In the nuclear domain, renewed national regulation can be a driver for unified safety and security considerations, as the example of STUK YVL guides suggest (cf. part A, §4.1.2); these guides set requirements to the nuclear power plant operators that only can be covered by seamless integration of safety and security experts. Other industries, e.g. transport the aviation domain, have privileged physical security (Prentice, 2002) and / or have been developing security standards for software-intensive systems, which cannot be termed as acceptable means of compliance (AMC), since there is no regulation to comply with. This situation is bound to change.

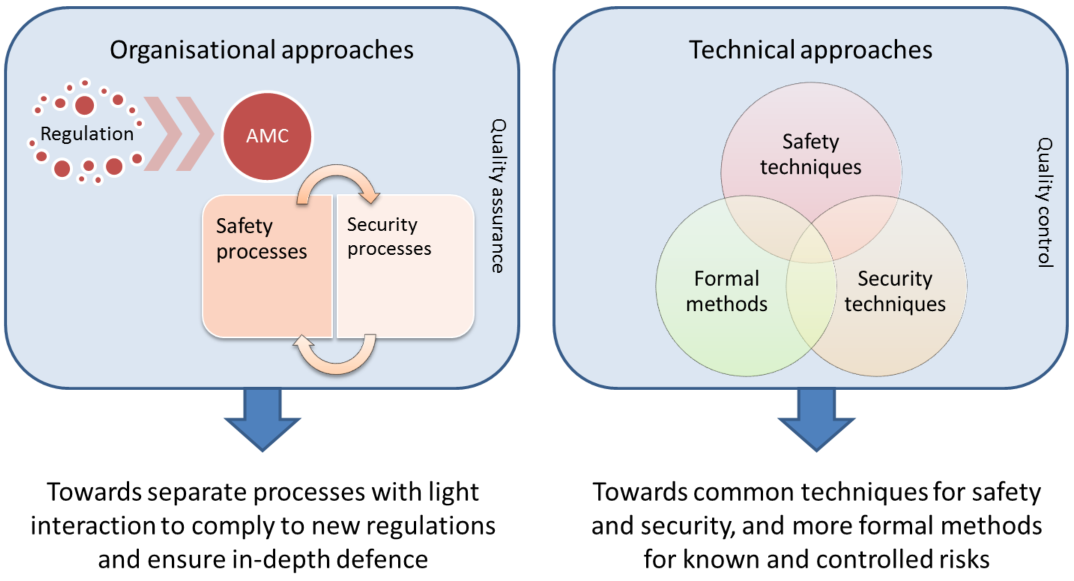

Organisational approaches
Regulation
AMC
Safety processes
Security processes
Quality assurance
Towards separate processes with light interaction to comply to new regulations and ensure in-depth defence
Technical approaches
Safety techniques
Formal methods
Security techniques
Quality control
Towards common techniques for safety and security, and more formal methods for known and controlled risks

Figure 55: Identified trends in safety and security engineering

Trends were a bit more difficult to establish. We have formulated two of them based on concordant events happening in multiple domains (e.g. aviation, electronics, nuclear), and on both side of the Atlantic:

- the safety communities thrive to maintain current organizational approaches as stable as possible, because regulations, acceptable means of compliance and standards have proven efficiency records and are extremely difficult to change, technically and / or politically (cf. Figure 55); some minor updates to the processes and methods are however necessary to ensure interaction points (SeSaMo, 2012), such as safety-aware security in (EUROCAE ED-202A, 2014) / (RTCA DO-326A, 2014), or security-aware safety in (S + IEC 61508, 2010); the safety communities seems to be moving away from revolutionising standard safety processes, e.g. with the obsolete (EUROCAE ED-202, 2010) / (RTCA DO-326, 2010), even if all individual members do not seem to adhere to this trend;
- the academic and industrial communities are adapting and extending existing techniques (Schmittner, et al., 2014b), architectures (Boettcher, et al., 2008) and tools (EURO-MILS EC FP7 Project, 2012), (Chapon, et al., 2012) to cover both safety and security properties; within this trend, the adoption and seamless integration of formal methods and tools (Garavel, et al., 2013), (Fisher, 2013) occupies a significant part (cf. Figure 55).

6 Recommendations for safety and security co-engineering

Our initial recommendations are based on some assumptions that have been derived from the state of the art (cf. Part A of this deliverable and its synthesis in §5). This section first exposes these assumptions and then proposes some way forward for safety and security co-engineering. Naturally, the proposed way forward echoes the initial motivations of this study, as presented in §1.3.

6.1 Assumptions

This section exposes three assumptions and the grounds that led us to believe these assumptions to be true.

Assumption n°1: **Industrial safety and security engineering processes / methods are difficult, and at best very slow, to change.**

The grounds for supporting that assumption are as follows:

- Industrial safety and security engineering processes / methods are defined in standards which are engineering specialty-specific[54]; for the safety specialty, they are moreover domain-specific[55]; they can also be customer RFT-related, legacy-related, proprietary-related, etc.
- The state of the art shows that numerous safety and security co-engineering processes have been proposed, but none have really emerged.

Assumption n°2: **Safety and security jargon is difficult, and at best very slow, to change**.

The grounds for supporting that assumption are as follows:

- There is no common glossary, even within a given engineering specialty.
- Safety and security jargon is engineering specialty-related, process-related, domain standard-related, customer RFT-related, legacy-related, proprietary-related, etc.
- Communities of specialty-experts are and remain essentially apart, even if some efforts exist to join those communities (cf. Part A, §6).

Assumption n°3: **Safety and security tools are diverse, but tend towards a formalisation of their conceptual data model**, in particular to suppress ambiguities & ensure coverage, to support analyses and to support interchange between tools.

The grounds for supporting that assumption are as follows:

- the (RTCA DO-178C, 2011) / (EUROCAE ED-12C, 2012) standard now recommends the use of formal methods;
- many techniques, e.g. Fault trees, Altarica, Attack Trees, SysML profile extensions, now support formal analyses;
- there are some initiatives to ensure interoperability between tools, which subsumes some form of formalisation of the exchanges, e.g. OpenPSA[56].

6.2 Proposals for safety and security co-engineering

The following proposals were exposed early in the MERgE project as elements of a research roadmap. Some leads were followed during the project, meaning that we are now in a position to provide some feedback of these early proposals, whilst other leads will need further work beyond the scope of the project.

6.2.1 Proposal n°1: a Common Model

The first proposal is artefact-related: **Intermediate safety and security work products can be shared between the two engineering specialties as long as the vernacular is maintained for each specialty**.

This proposal implies:

- the definition of a safety and security common work product model, hereafter called the “Common Model”;
- a mapping between specialty concepts and the Common Model concepts.

[54] Meaning safety-specific or security-specific.
[55] E.g. automotive, avionics, etc.
[56] Cf. http://open-psa.org/.

In this proposal (cf. Figure 56):

- all specialty concepts do not have to be mapped; e.g. the concept of "Target Level of Safety (TLS)" in the safety engineering specialty does not have a direct equivalent in the security engineering specialty;
- all concepts in the Common Model must have at least one counterpart in each engineering specialty;
- mapping may not always be a 1-to-1 mapping.

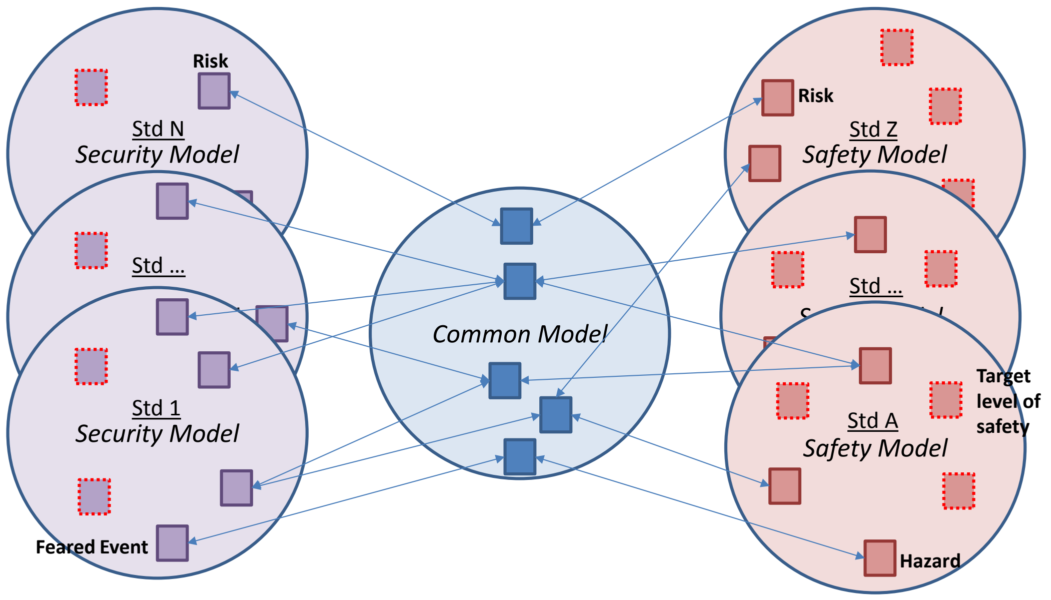


Figure 56: Proposal for a common work product model

In this proposal, even when specialty concepts are mapped, each engineering specialty retains its usual and / or standard term for the concept, and engineers proceed with their usual work, using their usual processes, methods and tools. E.g. when a safety expert creates a *Hazard*, it may be viewed, and potentially modified, as a *Feared Event* by security experts (cf. Figure 56).

When we proposed this Common Model, we identified the following major challenges related to the definition of a safety and security common work product model and the mapping of concepts:

- the coverage of multiple standards in each engineering specialty;
- the consideration of multiple design abstraction levels, in order to cover the complete system development lifecycle in a consistent way;
- the mapping at class level vs. a mapping at attribute level, e.g. a security risk may be mapped to a safety risk, but are all the attributes of a security risk identical to the attributes of a safety risk?
- the scope of the different engineering specialties is not exactly the same, e.g. security engineering is more concerned by the environment in which the system-under-study is operated than safety engineering; this raises the question about the relevance of integrating these elements inside the common model.

We started work on the Common Model by creating a taxonomy of terms present in the Common Criteria (ISO/IEC 15408-1, 2009) and, in a lesser measure, in the Functional Safety of E/E/PE Safety-Related Systems (S + IEC 61508, 2010). An extract of our work related to the Common Criteria standard is presented in Part A, §3.5 of this deliverable. The conceptual model for (ISO/IEC 15408-1, 2009) displayed approximately 200 classes, all related to each other through a complex set of associations and generalisation relationships. The safety standard offered a smaller set of concepts, but at this very detailed level, the mapping between the concepts of the two standards seamed tremendously complex.

The lesson learnt from this work is as follows. The establishment of a Common Model should be based on empirical studies, rather than on the systematic analysis of standards. This should ensure that the resulting generic pivot model is useful and manageable.

6.2.2 Proposal n°2: Independent Engineering Processes

The second proposal is process-related: **Work on common safety and security work products should be transparent for each specialty**, except in case of conflict / inconsistencies.

In this proposal, safety (resp. security) experts may independently design safety (resp. security) barriers to improve the system. However, some security barriers may have detrimental safety impacts and vice-versa. The co-engineering workbench must ensure that there is no endless engineering loop between conflicting design patterns.

When we proposed these Independent Engineering Processes, we identified the following major challenges for insuring the transparency between the safety and security engineering specialties:

- detecting cases of conflict / inconsistencies between updates by the different engineering specialties;
- convergence assurance and / or optimisation of the overall de-conflicting process;
- effects of the “safety-first” or “security-first” hypotheses on the detection and solving of conflicts.

Unfortunately, none of the MERgE test cases proposed conflicting safety and security requirements for us to experiment with. There are however some lessons learnt from this disappointing experience:

1) Obviously, safety and security requirements can be mutually reinforcing, independent, or conflicting. To our knowledge, there are neither post-mortem statistics stating the average percentage of reinforcing, independent, and conflicting requirements in a system, nor evaluations of how hard is the decision-making related to solving conflicting requirements, if any. These statistics and evaluations would be extremely useful to know if it is worthwhile working on engineering support for conflicting requirements, especially concerning safety and security requirements.
2) It is somehow artificial to limit the trade-off analysis between conflicting requirements to the sole plane of safety and security requirements, considering all other specialities as independent. Indeed, a security requirement may have a slight detrimental effect on safety that makes it an acceptable choice during a trade-off analysis, but it may in the same time have strong detrimental effects on usability, costs and/or performance, that make it an unacceptable choice. A trade-off approach needs to be comprehensive with respect to all relevant criteria.

Further details can be found in (Sébastien Madelénat, 2016).

6.2.3 Proposal n°3: Conditions for successful cross-fertilisation

The third proposal is tool-vendor oriented: **A new tooled-up approach may be acceptable by industry if it is an add-on to existing (standard) processes with added-value related to formal analyses, and without significant negative side-effects,** incl. extra workload.

In D3.4.1 (Faucogney, et al., 2014), we have shown that some cross-fertilisation between engineering specialties has already been successful, e.g. fault trees have given rise to attack trees. New ones are regularly being proposed, e.g. HazOp, Diagrammatical Misuse Cases (cf. Part A, §2), or SysML profiles (cf. §4). This proposal tries to explicit the criteria for a successful cross-fertilisation.

The conditions that we see to provide a successful new approach are:

- the proposed approach must have a proven record in at least one of the engineering specialty or in mainstream system engineering;
- the extension is optional, i.e. the technique may continue to be used as it has always been used prior to its extension;
- the extension has limited negative side-effects, in particular in terms of extra workload;
- the new approach brings more formalisation, allowing for formal analyses.

When we proposed these conditions for successful cross-fertilisation, we identified the following major challenges for the definition of new tooled-up approaches:

- choice of the most relevant abstraction level(s);
- analysis of the side-effects;
- definition of the supporting algorithms and / or tools;
- reconciliation of quantitative and qualitative approaches, e.g. to determine the probability of occurrence of feared events and / or hazards.

During our state of the art work (cf. Part A of this deliverable), we identified one tool that satisfies all of the above criteria, namely TTool/AVATAR. We believe that this tool has the potential for industrial adoption: this is why it was analysed and evaluated in detail, cf. §4, and demonstrated four times within the Thales Group during 2015. The tool still suffers from some scalability issues that we hope to solve early 2016.

In parallel, we defined our own safety and security engineering tool that also satisfies all of the above criteria: it implements a new Failure Modes, Effects and Criticality Analysis (FMECA) extended with security-related features (cf. §3).

7 References

25-356-SC Special Conditions: Boeing Model 787-8 Airplane; Systems and Data Networks Security-Isolation or Protection From Unauthorized Passenger Domain Systems Access [Online] // Federal Register. - Federal Aviation Administration, 01 02 2008. - 12 11 2014. - https://federalregister.gov/a/E7-25467. - 73 FR 27.

25-357-SC Special Conditions: Boeing Model 787-8 Airplane; Systems and Data Networks Security-Protection of Airplane Systems and Data Networks from Unauthorized External Access [Online] // Federal register. - Federal Aviation Administration, 28 12 2007. - 10 10 2014. - https://federalregister.gov/a/E7-25075. - E7-25075.

79 FR 60574 Request for Comment on Automotive Request for Comment on Automotive Security [Book Section] // Federal Register Volume 79, Issue 194 (October 7, 2014). - [s.l.] : Office of the Federal Register, National Archives and Records Administration, 2014. - 194 : Vol. 79. - FR Doc. 2014–23805.

Åkerberg Johan On Safe and Secure Communication in Process Automation [Report] : PhD. Thesis / School of Innovation, Design and Engineering. - Västerås (Sweden) : Mälardalen University Press Dissertations, 2011. - p. 57. - ISBN: 978-91-7485-039-0.

Altran Praxis SafSec Methodology, Issue 3.1 [Report] : Standard. - 2006. - S.P1199.50.2.

Alves-Foss Jim, Rinker Bob and Taylor Carol Towards Common Criteria Certification for DO-178B Compliant Airborne Software Systems, Comparing Evaluation Assurance Level 5 (EAL5) to DO178 [Online] // University of Idaho, Department of Computer Science, Jim Alves-Foss, Recent Publications and Presentations. - Center for Secure and Dependable Systems, 01 2002. - Draft. - 10 07 2014. - http://www2.cs.uidaho.edu/~jimaf/papers/compare02b.pdf. - Not releasable to the Defense Technical Information Center per DOD directive 3200.12..

Aoyama T. [et al.] A unified framework for safety and security assessment in critical infrastructures [Book Section] // Safety and security engineering V / ed. Garzia F., Brebbia C. A. and Guarascio M.. - [s.l.] : WIT Press, 2013. - Vol. 134. - DOI: 10.2495/SAFE130071.

Apvrille Ludovic [et al.] Secure automotive on-board electronics network architecture [Conference] // World Automotive Congress. - Budapest, Hungary : FISITA, 2010b.

Apvrille Ludovic and de Saqui-Sannes Pierre Static Analysis Techniques to Verify Mutual Exclusion Situations within SysML Models [Book Section] // SDL 2013: Model-Driven Dependability Engineering - Proceedings of the 16th International SDL Forum, June 26-28, 2013 / ed. Khendek Ferhat [et al.]. - Montreal : Springer Berlin Heidelberg, 2013. - 10.1007/978-3-642-38911-5_6.

Apvrille Ludovic and Knorreck Daniel UML for Embedded Systems, Laboratory Session Introduction [Report] : User Manual. - Sophia Antipolis : Telecom ParisTech, Eurecom. - p. 13. - http://soc.eurecom.fr/UMLEmb/labs/lab_fans_avatar_0.pdf.

Apvrille Ludovic and Roudier Yves Towards the Model-Driven Engineering of Secure yet Safe Embedded Systems [Conference] // Proceedings of 1st International Workshop on Graphical Models for Security (GraMSec) / ed. Kordy B., Mauw S. and Pieters W.. - Grenoble : Electronic Proceedings in Theoretical Computer Science, 2014. - Vol. 148. - pp. 15-30. - DOI: 10.4204/EPTCS.148.2.

Apvrille Ludovic AVATAR [Online] // TTOOL. - Institut Télécom, Télécom Paris Tech, 2015. - 26 02 2015. - http://ttool.telecom-paristech.fr/avatar.html.

Apvrille Ludovic Course on UML for Embedded Systems [Online]. - Telecom ParisTech, 2014. - 13 03 2015. - http://soc.eurecom.fr/UMLEmb/index.html.

Apvrille Ludovic et Roudier Yves SysML-Sec Attack Graphs: Compact Representations for Complex Attacks [Conférence] // Second International Workshop on Graphical Models for Security (GraMSec). - Verona : [s.n.], 2015. - p. 15. - (To be published).

ARINC 811 Commercial Aircraft Information Security Concepts of Operation and Process [Report] : Standard / Airlines Electronic Engineering Committee (AEEC). - Annapolis : Aeronautical Radio Incorporated (ARINC), 2005.

AVATAR Automated Verification of reAl Time softwARe (AVATAR) [Online] // TTOOL. - Institut Télécom, Télécom Paris Tech, 2015. - 26 02 2015. - http://ttool.telecom-paristech.fr/avatar.html.

Aven Terje A unified framework for risk and vulnerability analysis covering both safety and security [Book Section] // Proceedings of Reliability Engineering & System Safety. - 2007.

Aven Terje A unified framework for risk and vulnerability analysis covering both safety and security [Book Section]. - [s.l.] : IEEE, 2011. - 4th : Vol. 39. - DOI: 10.1109/EMR.2011.6093894.

Aven Terje Identification of safety and security critical systems and activities [Book Section] // Reliability Engineering & System Safety. - 2009. - DOI: 10.1016/j.ress.2008.04.001.

Avizienis Algirdas [et al.] Basic concepts and taxonomy of dependable and secure computing [Book Section] // IEEE Trans. Dependable Secur. Comput.. - 2004.

Axelrod C. Warren Applying Lessons from Safety-Critical Systems to Security-Critical Software [Conference] // IEEE Systems, Applications and Technology Conference (LISAT). - Farmingdale, NY, USA : [s.n.], 2011. - DOI: 10.1109/LISAT.2011.5784222.

Axelrod C. Warren Engineering Safe and Secure Software Systems [Book]. - [s.l.] : Artech House Publishers, 2012. - ISBN: 978-1-60807-473-0.

Axelrod C. Warren Managing the risks of cyber-physical systems [Book Section] // IEEE Systems, Applications and Technology Conference (LISAT). - Farmingdale : IEEE, 2013c. - DOI: 10.1109/LISAT.2013.6578215.

Axelrod C. Warren Securing Cyber-Physical Software [Online] // APPSEC USA. - 18-21 Nov. 2013b. - 05 May 2014. - http://2013.appsecusa.org/2013/wp-content/uploads/2013/12/APPSEC2013-Presentation-Final.ppt.

Babeshko E., Kharchenko V. and Gorbenko A. Applying F(I)MEA-technique for SCADA-Based Industrial Control Systems Dependability Assessment and Ensuring [Conference] // Third International Conference on Dependability of Computer Systems (DepCos-RELCOMEX). - Szklarska Poreba : IEEE, 2008. - pp. 309 - 315. - DOI: 10.1109/DepCoS-RELCOMEX.2008.23.

Banerjee Ayan [et al.] Ensuring Safety, Security, and Sustainabilityof Mission-Critical Cyber–Physical Systems [Journal] // Proceedings of the IEEE. - [s.l.] : IEEE, 2012. - 1 : Vol. 100. - pp. 283-299. - 10.1109/JPROC.2011.2165689.

Bezzateev Sergey, Voloshina Natalia and Sankin Petr Joint Safety and Security Analysis for Complex Systems [Conference] // 13th Conference of Open Innovations Association FRUCT. - Petrozavodsk, Russia : [s.n.], 2013.

Bieber Pierre [et al.] Security and Safety Assurance for Aerospace Embedded Systems [Book Section] // Embedded Real-Time Software and Systems (ERTS). - Toulouse : [s.n.], 2012.

Bieber Pierre and Brunel Julien From Safety Models to Security Models: Preliminary Lessons Learnt [Conference] // 1st International Workshop on the Integration of Safety and Security Engineering (ISSE), 33rd International Conference on Computer Safety, Reliability and Security (SafeComp) / ed. Bondavalli Andrea, Ceccarelli Andrea and Ortmeier Frank. - Florence : Springer, 2014. - pp. 269-281. - LNCS 8696. - DOI: 10.1007/978-3-319-10557-4.

Blanchet Bruno ProVerif: Cryptographic protocol verifier in the formal model [Online]. - INRIA. - 02 03 2015. - http://prosecco.gforge.inria.fr/personal/bblanche/proverif/.

Blanchet Bruno, Smyth Ben and Cheval Vincent ProVerif 1.89: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial [Report]. - Paris-Saclay : INRIA, 2014. - p. 113. - http://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf.

Blanquart Jean-Paul [et al.] Similarities and dissimilarities between safety levels and security levels [Online] // ERTS'2012. - 03 02 2012. - 16 05 2014. - http://www.erts2012.org/Default.aspx?Id=1050&Idd=1129. - 8A.2.

Bloomfield Richard [et al.] How secure is ERTMS? [Conference] // Workshop on Dependable and Secure Computing for Large-scale Complex Critical Infrastructures (DESEC4LCCI) / ed. Ortmeier Frank and Daniel Peter. - Herrenkrug : Springer, 2012. - Vol. 7613. - pp. 247-258. - http://openaccess.city.ac.uk/1522/1/How%20secure%20is%20ERTMS.pdf. - DOI: 10.1007/978-3-642-33675-1_22.

Bock Hans-Hermann [et al.] Towards an IT Security Protection Profile for Safety-Related Communication in Railway Automation [Conference] // 31st International Conference on Computer Safety, Reliability and Security (SafeComp) / ed. Ortmeier Frank and Daniel Peter. - Magdeburg : Springer Berlin Heidelberg, 2012. - Vol. 7612. - pp. 137-148. - DOI: 10.1007/978-3-642-33678-2_12.

Boettcher Carolyn [et al.] The MILS Component Integration Approach to Secure Information Sharing [Conference] // Proceedings o fthe 27th IEEE/AIAA Digital Avionics Systems Conference (DASC). - St. Paul, MN : IEEE, 2008. - DOI: 10.1109/DASC.2008.4702758.

Braband Jens IT security for functional safety in railway automation [Conference] // 1st Workshop on Safety and Security. - Kaiserslautern : [s.n.], 2014b. - Slides only.

Braband Jens Towards an IT Security Framework for Railway Automation [Conference] // Embedded Real-Time Software and Systems (ERTS2). - Toulouse, France : [s.n.], 2014.

Braband Jens Towards an IT security protection profile for safety-related communication in railway automation [Conference] // Embedded Real-Time Software and Systems (ERTS2). - Toulouse : [s.n.], 2014a.

Braber F. den [et al.] Model-based risk assessment in a component-based software engineering process: the CORAS approach to identify security risks [Book Section] // Business Component-Based Software Engineering / book auth. Barbier Franck. - 2003.

Brewer David F. C. Applying Security Techniques to Achieving Safety [Book Section] // Directions in Safety-Critical Systems, Proceedings of the First Safety-critical Systems Symposium, Bristol 9–11 February 1993 / book auth. Redmill Felix and Anderson Tom. - London : Springer, 1993. - DOI: 10.1007/978-1-4471-2037-7_16.

Brooke Phillip J. and Paige Richard F. Fault trees for security system design and analysis [Journal] // Computers & Security. - 2003. - 3 : Vol. 22. - pp. 256-264. - DOI: 10.1016/S0167-4048(03)00313-4.

Brostoff Sacha and Sasse M. Angela Safe and sound: a safety-critical approach to security [Book Section] // NSPW'01 Proceedings of the 2001 workshop on new security paradigms / ed. ACM. - New York : [s.n.], 2001. - DOI: 10.1145/508171.508178.

Brunel Julien [et al.] A Viewpoint-Based Approach for Formal Safety & Security Assessment of System Architectures [Conference] // 11th Workshop on Model Driven Engineering, Verification and Validation (MoDeVVa) / ed. Boulanger Frédéric, Famelis Michalis and Ratiu Daniel. - Valencia : [s.n.], 2014b. - pp. 39-48.

Brunel Julien [et al.] Formal Safety and Security Assessment of an Avionic Architecture with Alloy [Book Section] // 3rd International Workshop on Engineering Safety and Security Systems (ESSS). - Singapore : EPTCS, 2014a. - Vol. 150. - DOI: 10.4204/EPTCS.150.2.

Brunel Julien and Chemouil David Safety and Security Assessment of Behavioral Properties Using Alloy [Conference] // 2nd International workshop on the Integration of Safety and Security Engineering / ed. Koornneef F. and Gulijk C. van. - Delft : Springer International Publishing Switzerland, 2015. - Vol. LNCS 9338. - pp. 251–263. - DOI: 10.1007/978-3-319-24249-1 22.

BS EN 61508-1 Functional safety of electrical/ electronic/ programmable electronic safety-related systems - General requirements [Book]. - [s.l.] : British Standards Institution (BSI), 2002. - p. 68. - Withdrawn - replaced by BS EN 61508-1:2010. - ISBN: 0 580 32719 1.

Burns A., McDermid J. and Dobson J. On the meaning of safety and security [Book Section] // The Computer Journal - Special issue on safety and security parallel. - Oxford : Oxford University Press, 1992. - 1st : Vol. 35. - DOI: 10.1093/comjnl/35.1.3.

Carter Adele-Louise Safety-Critical versus Security-Critical Software [Conference] // 5th IET International Conference on System Safety. - Manchester, United Kingdom : [s.n.], 2010. - DOI: 10.1049/cp.2010.0814.

Casals Silvia Gil, Owezarski Philippe and Descargues Gilles Risk Assessment for Airworthiness Security [Book Section] // Computer Safety, Reliability, and Security, Lecture Notes in Computer Science. - 2012. - Vol. 7612. - DOI: 10.1007/978-3-642-33678-2_3.

CENELEC EN 20159 Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems [Report] : Standard. - [s.l.] : European Committee for Electro-technical Standardization, 2010. - Supersedes EN 50159-1:2001 and EN 50159-2:2001..

CENELEC EN 50128 Railway Applications: Software for Railway Control and Protection [Report] : Standard. - [s.l.] : European Committee for Electrotechnical Standardization, 1997.

Chapon Nicolas and Piètre-Cambacédès Ludovic Vers une ingénierie système intégrant sûreté et sécurité (in French) [Book Section] // Génie Logiciel. - 2012. - 100th.

Cherdantseva Yulia and Hilton Jeremy A Reference Model of Information Assurance & Security [Conference] // 8th International Conference on Availability, Reliability and Security (ARES). - Regensburg : IEEE, 2013. - DOI: 10.1109/ARES.2013.72.

Cimatti Alessandro [et al.] Combining MILS with Contract-Based Design for Safety and Security Requirements [Conference] // 2nd International workshop on the Integration of Safety and Security Engineering / ed. Koornneef F. and Gulijk C. van. - Delft : Springer International Publishing Switzerland, 2015. - Vol. LNCS 9338. - pp. 264–276. - DOI: 10.1007/978-3-319-24249-1 23.

Cockram Trevor J. and Lautieri Samantha R. Combining security and safety principles in practice [Conference] // 2nd Institution of Engineering and Technology International Conference on System Safety. - London : IET, 2007. - pp. 159 - 164. - ISBN: 978-0-86341-863-1.

Contini S., Cojazzi G.G.M. and Renda G. On the use of non-coherent fault trees in safety and security studies [Book Section] // Safety and Reliability for Managing Risk / book auth. Soares Guedes and Zio. - London : Taylor & Francis Group, 2006. - http://www.google.fr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0CDkQFjAC&url=http%3A%2F%2Fwww.dimat.unina2.it%2Fmarrone%2Fdwnld%2FProceedings%2FESREL%2F2006%2FPdf%2FS-341.pdf&ei=5QmUVOfGM8y9adyJgbgE&usg=AFQjCNHRv52Uh0W06G1Bbt_U-lG60IyScg&bvm=bv.8200133. - ISBN: 0-415-41620-5.

Corneillie Pierre [et al.] SQUALE Dependability Assessment Criteria [Report]. - [s.l.] : LAAS-CNRS, 1999. - 4th Edition. - ACTS95/AC097.

Cusimano John and Byres Eric Safety and Security: Two Sides of the Same Coin [Online]. - April 2010. - March 2014. - http://www.controlglobal.com/articles/2010/safetysecurity1004.

Czerny Barbara J. System Security and System Safety Engineering: Differences and Similarities and a System Security Engineering Process Based on the ISO 26262 Process Framework [Journal] // Journal of Passenger Cars – Electronic and Electrical Systems. - [s.l.] : SAE International, 2013. - 1 : Vol. 6. - DOI: 10.4271/2013-01-1419.

Daniel Hans Security in Safety Systems: the Need to Step beyond Traditional Engineering [Conference] // The Relationship between Safety and Security in Software-Based Systems, SafeComp Workshop. - 2008.

DARPA I2O HACMS High-Assurance Cyber Military Systems (HACMS) [Online] // Open Catalog / prod. Fisher Kathleen. - DARPA, 06 11 2014. - 21 11 2014. - http://www.darpa.mil/opencatalog/HACMS.html.

Daruwala Burzin [et al.] Threat Analysis for Hardware and Software Products Using HazOp [Book Section] // International Conference on Computational and Information Science (CIS'09). - Stevens Point : World Scientific and Engineering Academy and Society (WSEAS), 2009.

De Saqui-Sannes Pierre and Apvrille Ludovic AVATAR/TTool : un environnement en mode libre pour SysML temps réel [Conference] // Génie Logiciel. - [s.l.] : HAL, archives-ouvertes, 2011. - Vol. 98. - pp. 22-26. - In French. - hal-00667856.

Delange Julien Security and dependability integration for the construction of critical middleware (in French: Intégration de la sécurité et de la sûreté de fonctionnement dans la construction d'intergiciels critiques) [Report] : PhD Thesis / Laboratoire Traitement et Communication de l'Information, UMR 5141 ; Département informatique et réseau. - Paris : Ecole Nationale Supérieure des Télécommunications (TELECOM ParisTech), 2010. - p. 276. - pastel-00006301.

Deleuze Gilles [et al.] Are safety and security in industrial systems antagonistic or complementary issues? [Book Section] // 17th European safety and reliability conference (ESREL) / book auth. Martorell Sebastián, Guedes Soares Carlos and Barnett Julie. - Valencia : CRC Press, 2008. - http://hal-ineris.ccsd.cnrs.fr/ineris-00970394.

Derock A., Hebrard P. and Vallée F. Convergence of the latest standards addressing safety and security for information technology [Book Section] // On-line proceedings of Embedded Real Time Software and Systems (ERTS2 2010). - Toulouse, France : [s.n.], 2010.

Dewar Robert B. K. Safety and security: two sides of the same coin? [Conference] // The Relationship between Safety and Security in Software-Based Systems, SafeComp Workshop. - 2008.

Directive 95/46/EC Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data [Online] // EUR-Lex. - European Parliament, Council of the European Union, 24 10 1995. - 18 09 2015. - http://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:31995L0046.

D-MILS Distributed MILS for Dependable Information and Communication Infrastructures [Online]. - Scott Hansen, 2007. - 12 06 2015. - www.d-mils.org/.

Dolev Danny and Yao Andrew C. On the security of public key protocols [Journal] // IEEE Transactions on Information Theory / ed. IEEE. - 1983. - 2 : Vol. 29. - pp. 198-208. - DOI: 10.1109/TIT.1983.1056650.

Eames David Peter and Moffett Jonathan The Integration of Safety and Security Requirements [Book Section] // SAFECOMP '99 Proceedings of the 18th International Conference on Computer Computer Safety, Reliability and Security. - London : Springer-Verlag, 1999. - ISBN:3-540-66488-2 .

ECSS-Q-ST-30C Space product assurance - Dependability [Report] : Standard / ESA Requirements and Standards Division. - Noordwijk : European Cooperation on Space Standardization (ECSS), 2009. - p. 54.

ECSS-Q-ST-40C Space product assurance - Safety [Report] : Standard / ESA Requirements and Standards Division. - Noordwijk : European Cooperation on Space Standardization (ECSS), 2009. - p. 75.

ECSS-Q-ST-80C Space product assurance - Software product assurance [Report] : Standard / ESA Requirements and Standards Division. - Noordwijk : European Cooperation on Space Standardization (ECSS), 2009. - p. 113.

ETR 367 Telecommunications Security; Guidelines on the relevance of security evaluation to ETSI standards [Report] : ETSI Technical Report (ETR) 367. - Sophia Antipolis - Valbonne : European Telecommunications Standards Institute, 1997. - p. 21. - DTR/SEC-002701.

EU COM(2012) 11 final Reform of the data protection legal framework in the EU [Online] // European Commission - Justice - Data protection. - 01 2012. - 18 09 2015. - http://ec.europa.eu/justice/data-protection/reform/index_en.htm.

EUROCAE ED-12B Software Considerations in Airborne Systems and Equipment Certification [Report] : Standard. - [s.l.] : European Organisation for Civil Aviation Equipment, 1992. - Superseded by EUROCAE ED-12C:2012. - WG-12.

EUROCAE ED-12C Software Considerations in Airborne Systems and Equipment Certification [Report] : Standard. - [s.l.] : European Organisation for Civil Aviation Equipment, 2012. - WG-12.

EUROCAE ED-202 Airworthiness security process specification [Report]. - [s.l.] : European Organization for Civil Aviation Equipment (EUROCAE), 2010. - Superseded by EUROCAE ED-202A:2014. - WG-72.

EUROCAE ED-202A Airworthiness Security Process Specification [Report] : Standard. - [s.l.] : European Organization for Civil Aviation Equipment (EUROCAE), 2014.

EUROCAE ED-79A Guidelines for Development of Civil Aircraft and Systems [Rapport] : Standard. - [s.l.] : European Organisation for Civil Aviation Equipment, 2010.

EURO-MILS EC FP7 Project EURO-MILS [Online]. - 01 10 2012. - 16 09 2014. - http://www.euromils.eu/.

Fallière Nicolas Stuxnet Introduces the First Known Rootkit for Industrial Control Systems [Online] // Symantec Connect (Blog). - Symantec, 06 08 2010. - 24 11 2014. - http://www.symantec.com/connect/blogs/stuxnet-introduces-first-known-rootkit-scada-devices.

Faucogney Anthony and al. Report on open-issues in security and safety concern integration [Report] / ITEA2 – Project #11011. - [s.l.] : Multi-Concerns Interactions System Engineering (MERgE), 2014. - D3.4.1.

Favaro John and Stroud Robert ARTEMIS SESAMO Project: Work Achieved and Perspectives [Conference] // 1st International Workshop on the Integration of Safety and Security Engineering (ISSE), 33rd International Conference on Computer Safety, Reliability and Security (SafeComp). - Florence : [s.n.], 2014. - Introductory talk - Slides only.

Firesmith Donald G. Common concepts underlying safety, security, and survivability engineering [Report] / Software Engineering Institute ; Carnegie Mellon University. - 2003. - CMU/SEI-2003-TN-033.

Firesmith Donald G. Tutorial: Engineering safety- and security-related requirements for software-intensive systems [Conference] // 6th International Workshop on Software Engineering for Secure Systems (SESS'10) Workshop at the 32nd ICSE Conference. - Cape Town, South Africa : [s.n.], 2010.

Fisher Kathleen High Assurance Cyber Military Systems (HACMS): Making sure you are in control of your vehicle [Conference]. - [s.l.] : DARPA, 2013. - p. 33.

Förster Marc, Schwarz Reinhard and Steiner Max Integration of modular safety and security models for the analysis of the impact of security on safety [Report]. - [s.l.] : Fraunhofer IESE, 2010. - IESE-078.10/E.

Foster Nathalie Louise The application of software and safety engineering techniques to security protocol development [Report] : Thesis. - [s.l.] : University of York, 2002.

Fovino Igor Nai, Masera Marcelo and De Cian Alessio Integrating Cyber Attacks within Fault Trees [Journal] // Reliability Engineering and System Safety / ed. LTD ELSEVIER SCI. - 2009. - 9 : Vol. 94. - pp. 1394-1402. - 10.1016/j.ress.2009.02.020.

Fruth Jana and Nett Edgar Uniform Approach of Risk Communication in Distributed IT Environments Combining Safety and Security Aspects [Conference] // 1st International Workshop on the Integration of Safety and Security Engineering (ISSE), 33rd International Conference on Computer Safety, Reliability and Security (SafeComp) / ed. Bondavalli Andrea, Ceccarelli Andrea and Ortmeier Frank. - Florence : Springer, 2014. - pp. 289-300. - DOI: 10.1007/978-3-319-10557-4_32.

Garavel Hubert and Graf Susanne Formal Methods for Safe and Secure Computers Systems [Report] : Survey. - [s.l.] : Federal Office for Information Security, 2013. - p. 326. - BSI Study 875.

Gebauer Carsten Safety and security as drivers for future system development [Conference] // 1st Workshop on Safety and Security. - Kaiserslautern : [s.n.], 2014. - Slides only.

Gerhold Lars The Future of Research on Safety and Security in Germany - Results from an Explorative Delphi Study [Book Section] // Security in Futures – Security in Change, Proceedings of the Conference "Security in Futures – Security in Change", 3-4 June 2010, Turku, Finland / book auth. Auffermann Burkhard and Kaskinen Juha / ed. Auffermann Burkhard and Kaskinen Juha. - 2011. - FFRC eBOOK 5/2011.

Goertzel Karen Mercedes, Winograd Theodore and Hamilton Booz Allen Safety and Security Considerations for Component-Based Engineering of Software-Intensive Systems [Report]. - [s.l.] : Navy Software Process Improvement Initiative (SPII) and Department of Homeland Security, 2011.

Goertzel, Mercedes Karen and Feldman Larry Software Survivability: Where Safety and Security Converge [Book Section] // Proceedings of the American Institute of Aeronautics and Astronautics (AIAA) Infotech@Aerospace Conference. - Seattle : [s.n.], 2009.

Gorbenko Anatoliy [et al.] F(I)MEA-technique of Web Services Analysis and Dependability Ensuring [Book Section] // Rigorous Development of Complex Fault-Tolerant Systems, Lecture Notes in Computer Science / ed. Butler Michael [et al.]. - Berlin Heidelberg : Springer, 2006. - Vol. 4157. - DOI: 10.1007/11916246_8.

Green Hills Software Integrity Real-Time Operating System [Online]. - 2014. - 16 12 2014. - http://www.ghs.com/products/rtos/integrity.html.

Groll André [et al.] Next Generation of Automotive Security: Secure Hardware and Secure Open Platforms [Conference] // 17th ITS World Congress. - Busan, Korea : [s.n.], 2010.

Grøtan Tor Olav [et al.] The SeSa Method for Assessing Secure. Remote Access to Safety Instrumented [Report]. - Trondheim : Sintef, 2007. - p. 44. - https://www.sintef.no/globalassets/project/pds/reports/sintef-a1626-the-sesa-method-for-assessing-secure-remote-access-to-safety-instrumented-systems.pdf. - SINTEF A1626.

Gutgarts Peter B. and Temin Aaron Security-Critical versus Safety-Critical Software [Book Section] // Proceedings of IEEE International Conference on Technologies for Homeland Security (HST). - 2010. - DOI: 10.1109/THS.2010.5654973.

Hansen Kai Security attack analysis of safety systems [Book Section] // Proceedings of IEEE Conference on Emerging Technologies & Factory Automation (ETFA). - Mallorca : [s.n.], 2009. - DOI: 10.1109/ETFA.2009.5347258.

Helmer Guy [et al.] A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System [Journal] // Requirements Engineering. - London : Springer-Verlag, 2002. - 4 : Vol. 7. - pp. 207-220. - DOI: 10.1007/s007660200016.

Hessami Ali G. A systems framework for safety and security: The holistic paradigm [Section] // Systems Engineering. - [s.l.] : Wiley Periodicals, 2004. - 2nd : Vol. 7. - DOI: 10.1002/sys.10060.

Horn Marianne Developing safety-security critical systems: A prototype LEGO Mindstorm Detection System [Report] : Technical Report / Department of Computer and Information Science (IDI). - Trondheim : Norwegian University of Science and Technology (NTNU), 2005. - p. 82. - URL: http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2005/horn-fordyp05.pdf.

Hunter Bruce Integrating Safety and Security into the System Lifecycle [Conference] // Improving Systems and Software Engineering Conference (ISSEC) / ed. (ISSEC) Improving Systems and Software Engineering Conference. - Canberra : Eventcorp Pty Ltd, 2009. - pp. 147-158. - ISBN: 978-0-9807680-0-8.

Ibrahim Linda [et al.] Safety and Security Extensions for Integrated Capability Maturity Models [Report] / United States Federal Aviation Administration. - 2004.

ICAO ICAO Secretariat Study on the Safety and Security Aspects of Economic Liberalization [Report]. - [s.l.] : International Civil Aviation Organization, 2005.

IEC 31010 Risk management -- Risk assessment techniques [Report]. - [s.l.] : International Electrotechnical Commission, 2009. - p. 176. - ISO/TMBG.

IEC 60880 Software for computers in the safety systems of nuclear power stations [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 1986. - p. 133. - Superseded by IEC 60880 ed2.0 (2006-05).

IEC 61508-1 Functional safety of electrical / electronic / programmable electronic safety-related systems - Part 1: General requirements [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 1998. - p. 115. - (withdrawn). - Ed1.0.

IEC 61508-7 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 7: Overview of techniques and measures [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2000. - p. 229. - (withdrawn). - ISBN: 2-8318-5151-3.

IEC 62443-2-1 Industrial communication networks - Network and system security - Part 2-1: Establishing an industrial automation and control system security program [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2010. - TC/SC 65.

IEC 62443-3-3 Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2013.

IEC 62645 Nuclear power plants - Instrumentation and control systems - Requirements for security programmes for computer-based systems [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2014. - p. 93.

IEC/TR 62443-3-1 Industrial communication networks - Network and system security - Part 3-1: Security technologies for industrial automation and control systems [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2009.

IEC/TS 62443-1-1 Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2009. - TC/SC 65.

ISO 14971 Medical devices -- Application of risk management to medical devices [Report]. - [s.l.] : International Organization for Standardization, 2007. - p. 82. - Revises ISO 14971:2000. - ISO/TC 210.

ISO 25119-2 Tractors and machinery for agriculture and forestry -- Safety-related parts of control systems -- Part 2: Concept phase [Report]. - [s.l.] : International Organization for Standardization, 2010. - p. 37. - ISO/TC 23/SC 19.

ISO 26262-1 Road vehicles -- Functional safety -- Part 1: Vocabulary [Report] : Standard. - [s.l.] : International Organization for Standardization, 2011. - p. 23. - ISO/TC 22/SC 3.

ISO 26262-10 Road vehicles -- Functional safety -- Part 10: Guideline on ISO 26262 [Report] : Standard. - [s.l.] : International Organization for Standardization, 2012. - p. 89. - ISO/TC 22/SC 3.

ISO 31000 Risk management – Principles and guidelines [Report] : Standard. - [s.l.] : International Organization for Standardization, 2009. - p. 24. - ISO/TC 262.

ISO/IEC 15026 Information technology -- System and software integrity levels [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 1998. - Withdrawn. - ISO/IEC JTC 1/SC 7.

ISO/IEC 15026-2 Systems and software engineering -- Systems and software assurance -- Part 2: Assurance case [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 2011. - p. 10. - ISO/IEC JTC 1/SC 7.

ISO/IEC 15408-1 Information technology -- Security techniques -- Evaluation criteria for IT security -- Part 1: Introduction and general model [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 2009. - p. 64. - ISO/IEC JTC 1/SC 27.

ISO/IEC 17799 Information technology -- Security techniques -- Code of practice for information security management [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 2005. - Revised by: ISO/IEC 27002:2005. - ISO/IEC JTC 1/SC 27.

ISO/IEC 27005 Information technology -- Security techniques -- Information security risk management [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 2011. - p. 68. - ISO/IEC JTC 1/SC 27.

ISO/IEC 27005 Information technology -- Security techniques -- Information security risk management [Report] : Standard. - [s.l.] : International Standards Organization / International Electrotechnical Commission, 2008. - Withdrawn. - ISO/IEC JTC 1/SC 27.

ITSEC Information Technology Security Evaluation Criteria (ITSEC) - Harmonized Criteria of France, Germany, the Netherlands, the United Kingdom [Report] / Department of Trade and Industry. - London : Commission of the European Communities, 1991. - p. 164. - Withdrawn - Available: https://www.bsi.bund.de/cae/servlet/contentblob/471346/publicationFile/30220/itsec-en_pdf.pdf.

Jackson Daniel Software Abstractions: Logic, Language, and Analysis. [Book]. - [s.l.] : The MIT Press, 2006. - ISBN:0262101149.

Jackson Dave and Dobbing Brian Changing Regulation in Safety and Security – Implications and Opportunities [Conference] // The Relationship between Safety and Security in Software-Based Systems, SafeComp Workshop. - 2008.

Jalouneix Jean, Cousinou Patrick and Jean Couturier Denis Winter Approche comparative entre sûreté et sécurité nucléaires [Report] : Technical Report. - [s.l.] : Institut de Radioprotection et the Sûreté Nucléaire (IRSN), 2009. - p. 26. - [inFrench]. - IRSN 2009/117.

Johnson Chris W. Using Assurance Cases and Boolean Logic Driven Markov Processes to Formalise Cyber Security Concerns for Safety-Critical Interaction with Global Navigation Satellite Systems [Conference] // 4th Formal Methods for Interactive Systems Workshop / ed. Bowen J. and Reeves S.. - Limerick, Ireland : [s.n.], 2011. - URL: http://www.dcs.gla.ac.uk/~johnson/papers/FMIS2011/Chris.pdf.

Johnson Christopher W. CyberSafety: On the Interactions Between CyberSecurity and the Software Engineering of Safety-Critical Systems [Book Section] // Achieving System Safety / book auth. Dale C. and Anderson T.. - London : Springer Verlag, 2012. - Paper to acompany a keynote address, 20th Annual Conference of the UK Safety-Critical Systems Club. - ISBN: 978-1-4471-2493-1.

Johnson Roger G. Adversarial safety analysis: borrowing the methods of security vulnerability assessments [Journal] // Journal of Safety Research. - Amsterdam : Elsevier Science B.V., 2004. - 3 : Vol. 35. - pp. 245-248. - DOI: 10.1016/j.jsr.2004.03.013.

Jonsson Erland and Olovsson Tomas On the Integration of Security and Dependability in Computer Systems [Conference] // International Conference on Reliability, Quality Control and Risk Assessment (IASTED). - Washington DC : [s.n.], 1992. - pp. 93-97. - http://publications.lib.chalmers.se/records/fulltext/167782/local_167782.pdf. - ISBN: 0-88986-171-4.

Jonsson Erland Towards an integrated conceptual model of security and dependability [Conference] // First International Conference on Availability, Reliability and Security (ARES). - [s.l.] : IEEE, 2006. - pp. 646-653. - DOI: 10.1109/ARES.2006.138.

Joyce Jeff and Fabre Laurent Integration of security & airworthiness in the context of certification and standardization [Conference] // 1st workshop on the Integration of Safety and Security Engineering (ISSE). - Florence : [s.n.], 2014. - p. 18. - Invited talk - Slides only.

Katta Vikash and Stålhane Tor Traceability of Safety Systems: Approach, Meta-Model and Tool Support [Report] : Technical Report / OECD Halden Reactor Project. - Trondheim : Institute for Energy Technology (IET), 2013b. - Available upon request. - HWR-1053.

Katta Vikash, Raspotnig Christian and Stålhane Tor Requirements management in a combined process for safety and security assessments [Conference] // 8th International Conference on Availability, Reliability and Security (ARES). - Regensburg, Germany : [s.n.], 2013a.

Knorreck Daniel and Apvrille Ludovic TEPE: A SysML Language for Time-Constrained Property Modeling and Formal Verification [Conference] // Third IEEE International workshop UML and Formal Methods (UML&FM). - Shanghai : IEEE, 2010. - DOI:10.1145/1921532.1921556.

Kornecki Andrew J. and Liu Mingye Fault Tree Analysis for Safety/Security Verification in Aviation Software [Journal]. - [s.l.] : Electronics, 2013a. - 1 : Vol. 2. - pp. 41-56. - DOI:10.3390/electronics2010041.

Kornecki Andrew J., Subramanian Nary and Zalewski Janusz Studying Interrelationships of Safety and Security for Software Assurance in Cyber-Physical Systems: Approach Based on Bayesian Belief Networks [Conference] // Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (FedCSIS). - Kraków : IEEE, 2013b. - pp. 1381–1387.

Koscher Karl [et al.] Experimental Security Analysis of a Modern Automobile [Conference] // Symposium on Security and Privacy (SP). - Oakland, CA, USA : IEEE, 2010. - pp. 447 - 462. - ISBN: 978-1-4244-6894-2.

Kriaa Siwar [et al.] A survey of approaches combining safety and security for industrial control systems [Journal] // ScienceDirect Publication: Reliability Engineering & System Safety. - [s.l.] : Elsevier, 2015. - Vol. 139. - pp. 156-178. - http://freepaper.me/download/PDF/10.1016-J.RESS.2015.02.008.PDF?hash=MMMz7yMUAtoI5XReLMQNGw. - DOI: 10.1016/j.ress.2015.02.008 .

Kriaa Siwar [et al.] Comparing two approaches to safety and security modelling: BDMP technique and CHASSIS method [Conference] // OECD Halden Reactor Project, 37th Enlarged Halden Programme Group (EHPG) meeting. - Storefjell : [s.n.], 2013. - number C4.14.

Kriaa Siwar, Bouissou Marc and Laarouchi Youssef A Model Based Approach For SCADA Safety and Security Joint Modelling: S-cube [Conference] // IET Safety and Cyber-Security Conference. - Bristol : [s.n.], 2015b.

Labreuche Christophe and Lehuédé Fabien MYRIAD: a tool suite for MCDA [Conference] // 4th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT) / ed. Montseny E. and Sobrevilla P.. - Barcelona : [s.n.], 2005. - pp. 204-209. - http://www.eusflat.org/publications_proceedings_EUSFLAT-LFA_2005.php. - ISBN: 84-7653-872-3.

Lano Kevin, Clark David and Androutsopoulos Kelly Safety and Security Analysis of Object-Oriented Models [Book Section] // Computer Safety, Reliability and Security - Proceedings of 21st International SAFECOMP Conference, Catania, Italy, September 10–13, 2002 / ed. Heidelberg Springer Berlin. - 2002. - Vol. 2434. - DOI: 10.1007/3-540-45732-1_10.

Laprie Jean-Claude Dependability: Basic Concepts and Terminology [Book]. - Vienna : Springer, 1992. - Vol. 5 : pp. 3-245. - DOI: 10.1007/978-3-7091-9170-5.

Line Maria B. [et al.] Safety vs. Security? [Conference] // International Conference on Probabilistic Safety Assessment and Management (PSAM 8). - New Orleans, USA : [s.n.], 2006.

Luiijf Eric and Paske Bert Jan te Cyber Security of Industrial Control Systems [Conference] // Global Conference on CyberSpace (GCCS). - The Hague, The Netherlands : [s.n.], 2015. - https://www.gccs2015.com/sites/default/files/documents/Cyber%20Security%20of%20Industrial%20Control%20Systems%20GCCS2015.pdf.

Lynch J. A. Applying Safety Critical Systems Engineering Techniques to Secure Systems [Report]. - 2002. - p. 82. - http://www.google.fr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCMQFjAA&url=http%3A%2F%2Fwww-users.cs.york.ac.uk%2F~jac%2FPublishedPapers%2FJimLynch.doc&ei=DPSTVMXxH9OUav6dgIgI&usg=AFQjCNFHqroMkJZN_waDdyfIWwzHkHS5rQ&bvm=bv.82001339,d.d2.

Lynx Software Technologies [Online]. - 2015. - 06 03 2015. - http://www.lynx.com/.

Macher Georg [et al.] A Combined Safety-Hazards and Security-Threat Analysis Method for Automotive System [Conference] // 2nd International workshop on the Integration of Safety and Security Engineering / ed. Koornneef Floor and Gulijk Coen van. - Delft : Springer International Publishing Switzerland, 2015b. - Vol. LNCS 9338. - pp. 237–250. - DOI: 10.1007/978-3-319-24249-1 21.

Macher Georg [et al.] SAHARA: A Security-Aware Hazard and Risk Analysis Method [Conference] // Design, Automation & Test in Europe Conference & Exhibition (DATE). - Grenoble : IEEE, 2015a. - pp. 621-624. - ISBN: 978-3-9815-3704-8.

MAFTIA Malicious-and Accidental-Fault Tolerance for Internet Applications [Online] // LAAS. - IST MAFTIA Project n°11583, 01 01 2000. - 25 09 2014. - http://webhost.laas.fr/TSF/cabernet/maftia/.

Mattila Minna Different Views on Defining Safety, Security and Social Responsibility [Journal] // Interdisciplinary Studies Journal - Special Issue on Security, Safety and Social Responsibility / ed. Laakkonen Tarja, Paasonen Jyri and Mattila Minna. - Helsinki : Prima Oy, 2013. - 1 : Vol. 3. - pp. 7-20. - ISSN: 1799-2710.

Mazzini Silvia [et al.] Security and Safety Modelling in Embedded Systems [Conference] // Embedded Real Time Software and Systems (ERTS). - Toulouse : [s.n.], 2014.

Mc Guire Nicholas Utilizing security methods of FLOSS GPOS for safety [Conference] // Embedded World Exhibition & Conference. - Nürnberg : [s.n.], 2011.

MERgE [Online] // Multi-Concerns Interactions System Engineering (MERgE) Project - Safety & Security / prod. 6 ITEA 2 call. - 2012. - 27 11 2014. - http://www.merge-project.eu/. - https://itea3.org/project/merge.html. - 11011.

MODSafe Modular Urban Transport Safety and Security Analysis [Online]. - EU FP7, 01 09 2008. - 21 05 2014. - http://www.modsafe.eu/.

Monakova Ganna, Brucker Achim D. and Schaad Andreas Security and safety of assets in business processes [Conference] // 27th Annual ACM Symposium on Applied Computing. - New York, NY, USA : Association for Computing Machinery, 2012. - pp. 1667-1673 . - DOI: 10.1145/2245276.2232045.

Müller Kevin [et al.] MILS-Based Information Flow Control in the Avionic Domain: a Case Study on Compositional Architecture and Verification [Conference] // 31st Digital Avionics Systems Conference (DASC). - Williamsburg : IEEE, 2012. - pp. 1-13. - DOI: 10.1109/DASC.2012.6382411.

Müller Kevin [et al.] MILS-related information flow control in the avionic domain: A view on security-enhancing software architectures [Conference] // 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W). - Boston, MA : IEEE, 2012b. - pp. 1-6. - 10.1109/DSNW.2012.6264665.

Müller Kevin [et al.] On MILS I/O Sharing Targeting Avionic Systems [Conference] // 10th European Dependable Computing Conference (EDCC). - Newcastle : IEEE, 2014. - pp. 182-193. - 10.1109/EDCC.2014.35.

Murdoch John [et al.] Security Measurement [Report] : White-Paper / Safety & Security Technical Working Group (TWG). - [s.l.] : Practical Software and Systems Measurement (PSM), 2006. - p. 67. - v3.0.

Netkachova Kateryna [et al.] Security-Informed Safety Case Approach to Analysing MILS Systems [Conference] // 1st International Workshop on MILS: Architecure and Assurance for Secure Systems. - Amsterdam, The Netherlands : [s.n.], 2015. - http://mils-workshop-2015.euromils.eu/.

Nicol David M., Sanders William H. and Trivedi Kishor S. Model-based evaluation: from dependability to security [Journal] // IEEE Transactions on Dependable and Secure Computing. - [s.l.] : IEEE, 2004. - 1 : Vol. 1. - pp. 48-65. - DOI: 10.1109/TDSC.2004.11.

Nielson Hanne Riis and Nielson Flemming Safety versus Security in the Quality Calculus [Book Section] // Theories of Programming and Formal Methods, Lecture Notes in Computer Science / book auth. Liu Zhiming, Woodcock Jim and Zhu Huibiao. - Berlin Heidelberg : Springer, 2013. - Vol. 8051. - DOI: 10.1007/978-3-642-39698-4_18.

NIST SP 800-53 Security and Privacy Controls for Federal Information Systems Federal Information Systems, Special Publication 800-53, Revision 4 [Report] / Computer Security Division ; Information Technology Laboratory. - Gaithersburg : National Institute of Standards and Technology, 2013. - p. 460. - http://dx.doi.org/10.6028/NIST.SP.800-53r4.

NIST SP 800-82 Guide to Industrial Control Systems (ICS) Security, Special Publication 800-82, Revision 1 [Report] : Standard / Computer Security Division ; Information Technology Laboratory. - Gaithersburg : National Institute of Standards and Technology, 2013. - p. 170. - http://dx.doi.org/10.6028/NIST.SP.800-82r1.

Nordland Odd Some Security Aspects in Safety-Related Systems [Conference] // The Relationship between Safety and Security in Software-Based Systems, SafeComp Workshop. - 2008.

Novak Thomas, Treytl Albert and Palensky Peter Common Approach to Functional Safety and System Security in Building Automation and Control Systems [Book Section] // Proceedings of Conference on Emerging Technologies and Factory Automation (ETFA). - [s.l.] : IEEE, 2007.

Olive Michael L., Oishi Roy T. and Arentz Stephen Commercial Aircraft Information Security — An Overview of ARINC Report 811 [Conference] // 25th Digital Avionics Systems Conference (DASC). - Portland : IEEE, 2006. - pp. 1-12. - DOI: 10.1109/DASC.2006.313761.

OMG SACM Structured Assurance Case Meta-model [Online] // Object Management Group. - 2 2013. - 23 05 2014. - http://www.omg.org/spec/SACM.

OMG SysML OMG Systems Modeling Language v1.3 [Report] = OMG SysML™ : Standard. - [s.l.] : Object Management Group (OMG), 2012. - p. 271. - formal/12-06-01.

Pan Dong-bo and Huang Wei Operation of Functional Safety and Security [Conference] // 2nd IEEE Conference on Industrial Electronics and Applications (ICIEA). - [s.l.] : IEEE, 2007b. - pp. 1318 - 1322 . - DOI: 10.1109/ICIEA.2007.4318619.

Pan Dong-bo and Liu Feng Influence between Functional Safety and Security [Conference] // 2nd IEEE Conference on Industrial Electronics and Applications (ICIEA). - [s.l.] : IEEE, 2007a. - pp. 1323 - 1325. - DOI: 10.1109/ICIEA.2007.4318620.

Paul Stéphane and Rioux Laurent Over 20 Years of Research in Cybersecurity and Safety Engineering: a short Bibliography [Conference] // 6th International Conference on Safety and Security Engineering (SAFE). - Opatija : [s.n.], 2015. - p. 15.

Paul Stéphane On the Meaning of Security for Safety (S4S) [Conference] // 6th International Conference on Safety and Security Engineering (SAFE). - Opatija : [s.n.], 2015.

Paulitsch Michael [et al.] Evidence-Based Security in Aerospace: From Safety to Security and Back Again [Conférence] // IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW). - [s.l.] : IEEE, 2012. - pp. 21-22. - DOI: 10.1109/ISSREW.2012.37.

Pedroza Gabriel, Apvrille Ludovic and Knorreck Daniel AVATAR: A SysML environment for the formal verification of safety and security properties [Conference] // 11th International Conference on New Technologies of Distributed Systems (NOTERE). - Paris : IEEE, 2011. - pp. 1-10. - DOI: 10.1109/NOTERE.2011.5957992.

Pfitzmann Andreas Why Safety and Security Should and Will Merge, Volume [Book Section] // Computer Safety, Reliability, and Security, Lecture Notes in Computer Science. - 2004. - Vol. 3219. - DOI: 10.1007/978-3-540-30138-7_1.

Pieters Wolter [et al.] Reconciling malicious and accidental risk in cyber security [Journal] // Journal of Internet Services and Information Security. - 2014. - 2 : Vol. 4. - pp. 4-26. - ISSN: 2182-2077 .

Piètre-Cambacedes Ludovic and Bouissou Marc Cross-fertilizations between safety and security engineering [Journal] // Reliability Engineering & System Safety. - [s.l.] : Elsevier B.V., 2013a. - Vol. 110. - pp. 110–126. - DOI: 10.1016/j.ress.2012.09.011.

Piètre-Cambacédès Ludovic and Bouissou Marc Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes), IEEE International Conference Systems Man and Cybernetics (SMC) [Book Section]. - Istanbul : [s.n.], 2010. - DOI: 10.1109/ICSMC.2010.5641922.

Piètre-Cambacédès Ludovic and Chaudet Claude Disentangling the relations between safety and security [Book Section] // AIC'09 Proceedings of the 9th WSEAS international conference on Applied informatics and communications. - Stevens Point : [s.n.], 2009. - ISBN: 978-960-474-107-6.

POK Community Home page [Online] // A Partioned Operating System (POK). - 31 01 2011. - 04 07 2014. - http://pok.tuxfamily.org/.

Prentice Stephen P. Safety Vs. Security: can we afford both? [Online] // AviationPros. - 01 04 2002. - 12 05 2015. - http://www.aviationpros.com/article/10387597/safety-vs-security-can-we-afford-both.

QNX QNX Hypervisor [Online]. - 2015. - 06 03 2015. - http://www.qnx.com/products/hypervisor/index.html.

Ramirez Adrian Garcia [et al.] On Two Models of Noninterference: Rushby and Greve, Wilding, and Vanfleet [Conference] // 33rd International Conference (SAFECOMP) / ed. Bondavalli Andrea and Giandomenico Felicita Di. - Florence : Springer International Publishing, 2014. - Vol. 8666. - pp. 246-261. - DOI: 10.1007/978-3-319-10506-2_17.

Raspotnig Christian and Opdahl Andreas L. Comparing risk identification techniques for safety and security requirements [Journal] // Journal of Systems and Software. - 2013a. - 4 : Vol. 86. - pp. 1124 – 1151. - DOI: 10.1016/j.jss.2012.12.002.

Raspotnig Christian and Opdahl Andreas L. Improving security and safety modelling with failure sequence diagrams [Book Section] // International Journal of Secure Software Engineering (IJSSE). - 2012a. - DOI: 10.4018/jsse.2012010102.

Raspotnig Christian Requirements for safe and secure information systems [Rapport] : Thesis / University of Bergen. - 2014.

Raspotnig Christian, Karpati Peter and Katta Vikash A Combined Process for Elicitation and Analysis of Safety and Security Requirements [Book Section] // Enterprise, Business- Process and Information Systems Modeling (EMMSAD), Lecture Notes in Business Information Processing / book auth. Bider I. [et al.]. - [s.l.] : Springer Berlin Heidelberg, 2012b. - Vol. 113. - DOI: 10.1007/978-3-642-31072-0_24.

Reichenbach Frank [et al.] A pragmatic approach on combined safety and security risk analysis [Book Section] // IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW). - Dallas : IEEE, 2012. - DOI: 10.1109/ISSREW.2012.98.

Ridgway John Achieving Safety through Security Management [Book Section] // The Safety of Systems / ed. Redmill Felix and Anderson Tom. - London : Springer, 2007. - DOI: 10.1007/978-1-84628-806-7_1.

Roth Michael and Liggesmeyer Peter Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees [Conference] // ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems (DECS) / ed. Roy Matthieu. - Toulouse : [s.n.], 2013. - hal-00848640.

Rowe Jayson Software Security & Design Assurance [Conference] // Design & Manufacture Seminar. - [s.l.] : Civil Aviation Safety Authority, Australian Goverment, 2013. - p. 35. - Slides only. - http://www.casa.gov.au/wcmswr/_assets/main/lib100210/d1t05.pdf.

RTCA DO-178B Software Considerations in Airborne Systems and Equipment [Report] : Standard. - Washington : Radio Technical Commission for Aeronautics, 1992. - Not superseded by RTCA DO-178C:2011. - SC-167.

RTCA DO-178C Software Considerations in Airborne Systems and Equipment [Report] : Standard. - Washington : Radio Technical Commission for Aeronautics, 2011. - p. 144. - RTCA DO-178C does not superseed RTCA DO-178B:1992. - SC-205.

RTCA DO-326 Airworthiness Security Process Specification [Report] : Standard. - [s.l.] : Radio Technical Commission for Aeronautics, 2010. - Superseded by RTCA DO-326A. - SC-216.

RTCA DO-326A Airworthiness Security Process Specification [Report] : Standard. - Washington : Radio Technical Commission for Aeronautics (RTCA), 2014. - p. 88. - SC-216.

Rushby John Critical properties: survey and taxonomy [Report] / Computer Science Laboratory ; SRI International. - Menlo Park : [s.n.], 1994. - CSL-93-01.

Rushby John Kernels for Safety? [Book Section] // Safe and Secure Computing Systems / book auth. Anderson T.. - [s.l.] : Blackwell Scientific Publications, 1989.

Rushdi Ali Muhammad and Ba-Rukab Omar M. A doubly-stochastic fault-tree assessment of the probabilities of security breaches in computer systems [Book Section] // Proceedings of the 2nd Saudi Science Conference. - 2004. - Vol. 4.

Rushdi Ali Muhammad and Ba-Rukab Omar M. Fault-tree modelling of computer system security [Journal] // International Journal of Computer Mathematics. - 2005. - Vol. 82. - pp. 805-819. - DOI: 10.1080/00207160412331336017.

S + IEC 61508 Functional safety of electrical / electronic / programmable electronic safety-related systems [Report] : Standard. - [s.l.] : International Electrotechnical Commission, 2010. - p. 1000. - Ed2.0.

Sadvandi Sara, Chapon Nicolas and Piètre-Cambacédès Ludovic Safety and Security Interdependencies in Complex Systems and SoS: Challenges and Perspectives [Book Section] // Complex Systems Design and Management (CSDM) / book auth. Hammami O., Krob D. and Voirin J.-L.. - [s.l.] : Springer Berlin Heidelberg, 2012. - DOI: 10.1007/978-3-642-25203-7_16.

SAE ARP 4754A Guidelines for Development of Civil Aircraft and Systems [Report] : Standard / Aerospace Recommended Practice (ARP). - [s.l.] : Society of Automotive Engineers (SAE) International, 2010.

Saglietti Francesca Common Analysis and Verification Techniques for Safety- and Security- Critical Software Systems [Conference] // The Relationship between Safety and Security in Software-Based Systems, SafeComp Workshop. - 2008.

Sallhammar Karin, Helvik Bjarne E. and Knapskog Svein J. Towards a Stochastic Model for Integrated Security and Dependability Evaluation [Conference] // First International Conference on Availability, Reliability and Security. - Washington : IEEE, 2006. - pp. 156-165. - DOI: 10.1109/ARES.2006.137.

Schmittner Christoph [et al.] A Case Study of FMVEA and CHASSIS as Safety and Security Co-Analysis Method for Automotive Cyber-physical Systems [Conference] // 1st ACM Workshop on Cyber-Physical System Security (CPSS). - New York : ACM, 2015a. - pp. 69-80 . - DOI: 10.1145/2732198.2732204.

Schmittner Christoph [et al.] Security Application of Failure Mode and Effect Analysis (FMEA) [Conference] // 33rd International Conference on Computer Safety, Reliability and Security (SafeComp) / ed. Bondavalli Andrea and Giandomenico Felicita Di. - Florence : Springer, 2014b. - DOI: 10.1007/978-3-319-10506-2_21.

Schmittner Christoph and Ma Zhendong Towards a Framework for Alignment Between Automotive Safety and Security Standards [Conference] // EWICS/ERCIM/ARTEMIS Dependable Cyber-physical Systems and Systems-of-Systems Workshop (DECSoS) / ed. Koornneef Floor and Gulijk Coen van. - Delft : Springer International Publishing Switzerland, 2015b. - Vol. LNCS 9338. - pp. 133–143. - DOI: 10.1007/978-3-319-24249-1 12.

Schmittner Christoph, Ma Zhendong and Gruber Thomas Standardization Challenges for Safety and Security of Connected, Automated and Intelligent Vehicles [Conference] // 3rd International Conference on Connected Vehicles & Expo (ICCVE). - Vienna : [s.n.], 2014c.

Schmittner Christoph, Ma Zhendong and Smith Paul FMVEA for Safety and Security Analysis of Intelligent and Cooperative Vehicles [Conference] // 1st International Workshop on the Integration of Safety and Security Engineering (ISSE), 33rd International Conference on Computer Safety, Reliability and Security (SafeComp) / ed. Bondavalli Andrea, Ceccarelli Andrea and Ortmeier Frank. - Florence : Springer, 2014a. - pp. 282–288. - LNCS 8696. - DOI: 10.1007/978-3-319-10557-4_31.

Schneider Daniel Runtime certification of safety and security in cyber-physical system [Conference] // 1st Workshop on Safety & Security. - Kaiserslautern : [s.n.], 2014. - Slides only..

Schoitsch Erwin Design for safety and security of complex embedded systems: a unified approach [Book Section] // Cyberspace Security and Defense: Research Issues, NATO Science Series II: Mathematics, Physics and Chemistry / book auth. Kowalik J., Gorski J. and Sachenko A.. - [s.l.] : Springer Netherlands, 2005. - Vol. 196. - DOI: 10.1007/1-4020-3381-8_9.

Schoitsch Erwin Safety and security – what about a joint process? [Conference] // 1st Workshop on Safety & Security. - Kaiserslautern : [s.n.], 2014. - p. 37. - Slides only.

Schwarz Reinhard My thoughts on safety and security metrics [Conference] // 1st IESE Workshop on Safety and Security. - Kaiserslautern : [s.n.], 2014. - Slides only.

Sébastien Madelénat Christophe Labreuche, Jérôme Le Noir, Grégory Gailliard Comparing several candidate architectures (variants) : An Industrial Case Study [Conference] // Embedded Real Time Software and Systems (ERTS²). - Toulouse, France : [s.n.], 2016. - p. 10. - http://www.erts2016.org/programme-thursday.html.

SEISES Cooperative Projects [Online] // Aerospace Valley. - FUI, 2008. - 21 05 2014. - http://www.aerospace-valley.com/les-projets?keywords=seises. - In French.

SeSaMo D2.1 Specification of Safety and Security Mechanisms [Report]. - [s.l.] : Security and Safety Modelling, Artemis JU Project Grant Agreement no.: 295354, 2013.

SeSaMo D3.1 Specification of Safety and Security Analysis and Assessment Techniques [Report]. - [s.l.] : Security and Safety Modelling, Artemis JU Project Grant Agreement no.: 295354, 2013.

SeSaMo D4.1 Integrated Design and Evaluation Methodology [Report]. - [s.l.] : Security and Safety Modelling, Artemis JU Project Grant Agreement no.: 29535, 2014.

SeSaMo Security and Safety Modelling [Online]. - 2012. - 20 05 2014. - http://sesamo-project.eu/.

Simpson Andrew, Woodcock Jim and Davies Jim Safety through Security [Book Section] // IWSSD'98 Proceedings of the 9th international workshop on Software specification and design / ed. Society IEEE Computer. - Washington : [s.n.], 1998. - ISBN:0-8186-8439-9.

Sindre Guttorm A look at misuse cases for safety concerns [Book Section] // Situational Method Engineering: Fundamentals and Experiences / book auth. Ralyt'e9 J., Brinkkemper S. and Henderson-Sellers B. / ed. Boston Springer. - [s.l.] : IFIP International Federation for Information Processing, 2007. - Vol. 244. - DOI: 10.1007/978-0-387-73947-2_20.

Smith J., Russell S. and Looi M. Security as a Safety Issue in Rail Communications [Book Section] // Proceedings of SCS'03, 8th Australian Workshop on Safety Critical Systems and Software, Canberra, October 9-10, 2003 / ed. Australian Computer Society Inc.. - Darlinghurst : ACM, 2003. - Vol. 33. - ISBN:1-920-68215-5.

Soja Richard Automotive Security: From Standards to Implementation [Report] : White paper. - [s.l.] : Freescale, 2014. - URL: http://cache.freescale.com/files/automotive/doc/white_paper/AUTOSECURITYWP.pdf. - AUTOSECURITYWP REV 1.

Sommerville Ian An Integrated Approach to Dependability Requirements Engineering [Book Section] // Current Issues in Safety-Critical Systems / book auth. Redmill Felix and Anderson Tom / ed. Springer. - London : [s.n.], 2003. - DOI: 10.1007/978-1-4471-0653-1_1.

Sørby Karine Relationship between security and safety in a security-safety critical system: Safety consequences of security threats [Report] : Master Thesis. - Trondheim, Norway : Norwegian University of Science and Technology (NTNU), 2003. - p. 185.

Srivatanakul Thitima Security Analysis with Deviational Techniques [Report] : PhD. Thesis. - York : University of York, Department of Computer Science, 2005. - p. 279.

Srivatanakul Thitima, Clark John A. and Polack Fiona Effective Security Requirements Analysis: HazOp and Use Cases [Book Section] // Information Security, Lecture Notes in Computer Science / book auth. Zhang K. and Zheng Y.. - Berlin / Heidelberg : Springer, 2004. - Vol. 3225.

Stålhane Tor and Sindre Guttorm Safety Hazard Identification by Misuse Cases: Experimental Comparison of Text and Diagrams [Book Section] // Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science / book auth. Czarnecki K. [et al.]. - [s.l.] : Springer Berlin Heidelberg, 2008. - Vol. 5301.

Stavridou V. and Dutertre B. From Security to Safety and Back [Book Section] // Computer Security, Dependability and Assurance: From needs to Solutions. Proceedings / ed. IEEE. - York : [s.n.], 1998. - DOI: 10.1109/CSDA.1998.798365.

Steiner Max and Liggesmeyer Peter Combination of Safety and Security Analysis - Finding Security Problems that Threaten the Safety of a System [Conference] // ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems (DECS), 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP). - 2013.

Stephenson Peter Information security in 2014: Another year of big events [Online] // Reviews. - SC Magazine, 8 12 2014. - 12 12 2014. - http://www.scmagazine.com/information-security-in-2014-another-year-of-big-events/article/384497/.

Stoneburner G. Toward a Unified Security-Safety Model [Book Section]. - [s.l.] : IEEE Computer, 2006. - Vol. 39.

Stumpf Frederic CycurHSM: An Automotive-qualified Software Stack for Hardware Security Modules [Report] : White paper. - Stuttgart, Germany : Escrypt, 2013. - p. 9. - URL: https://www.escrypt.com/fileadmin/escrypt/pdf/CycurHSM-Whitepaper.pdf.

Subramanian Nary and Zalewski Janusz Assessment of Safety and Security of System Architectures for Cyberphysical Systems [Conference] // International Systems Conference (SysCon). - Orlando, FL : IEEE, 2013. - pp. 634 - 641. - DOI: 10.1109/SysCon.2013.6549949.

Subramanian Nary and Zalewski Janusz Quantitative Assessment of Safety and Security of System Architectures for Cyberphysical Systems Using the NFR Approach [Journal] // IEEE Systems Journal / ed. IEEE. - 09 01 2014. - 99 : Vol. PP. - pp. 1-13. - DOI: 10.1109/JSYST.2013.2294628.

Sun Mu [et al.] Addressing Safety and Security Contradictions in Cyber-Physical Systems [Conference] // 1st Workshop on Future Directions in Cyber-Physical Systems Security (CPSSW). - Newark : US Department of Homeland Security, 2009. - http://cimic.rutgers.edu/positionPapers/cpssecurity09_MuSun.pdf.

Sysgo PikeOS Hypervisor [Online]. - 2014. - 16 12 2014. - http://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/.

Taguchi Kenji, Souma Daisuke and Nishihara Hideaki Safe & Sec Case Patterns [Conference] // 3rd International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE) / ed. Koornneef Floor and Gulijk Coen van. - Delft : Springer International Publishing Switzerland, 2015. - Vol. LNCS 9338. - pp. 27–37. - DOI: 10.1007/978-3-319-24249-1 3.

Taylor Carol, Alves-Foss Jim and Rinker Bob Merging Safety and Assurance: the Process of Dual Certification of Software [Conference] // Software Technology Conference. - 2002b.

Taylor Carol, Alves-Foss Jim and Rinker Bob Towards Common Criteria Certification for DO-178B: Executive Summary [Online] // University of Idaho, Department of Computer Science, Jim Alves-Foss, Recent Publications and Presentations. - Center for Secure and Dependable Systems, 03 2002a. - 10 07 2014. - http://www2.cs.uidaho.edu/~jimaf/papers/compare02a.pdf.

Tiwari Ashish [et al.] Safety Envelope for Security [Conference] // 3rd international conference on High Confidence Networked Systems (HiCoNS). - Berlin : ACM Digital Library, 2014. - pp. 85-94. - http://www.csl.sri.com/users/tiwari/papers/hicons14.pdf. - DOI: 10.1145/2566468.2566483.

Tverdyshev Sergey MILS – Architecture for Safety and Security [Conference] // 1st workshop on safety and security. - Kaiserslautern : [s.n.], 2014. - Slides only.

UP4ALL Design Verification for Embedded Systems [Online]. - UP4ALL International AB. - 02 03 2015. - http://www.uppaal.com/index.php.

Vogt Roland Safety / security conflicts in large system architectures [Conference] // 1st Workshop on Safety and Security. - Kaiserslautern : [s.n.], 2014. - Slides only.

Vouk Mladen A. Differences and Similarities between Software Reliability and Software Security Engineering [Conference] // Software Reliability in 2013: Theory & Practice. - Levallois-Perret : IEEE-RS, 2013.

Ward David, Ibarra Ireri and Ruddle Alastair Threat Analysis and Risk Assessment in Automotive Cyber Security [Conference] // SAE World Congress & Exhibition. - [s.l.] : SAE Int., 2013. - DOI:10.4271/2013-01-1415.

Wiander Timo Positive and Negative Findings of the ISO/IEC 17799 Framework [Conference] // Proceedings of the 18th Australasian Conference on Information Systems (ACIS). - Toowoomba : AIS Electronic Library (AISeL), 2007. - Paper 75..

Wind River VxWorks [Online]. - 2015. - 06 03 2015. - http://www.windriver.com/products/vxworks/.

Winther Rune Qualitative and Quantitative Analysis of Security in Safety and Reliability Critical Systems [Book Section] // Probabilistic Safety Assessment and Management / ed. Schmocker Cornelia Spitzer . Ulrich and Dang Vinh N.. - London : Springer, 2004. - Vol. 6. - DOI: 10.1007/978-0-85729-410-4_377.

Winther Rune, Johnsen Ole-Arnt and Gran Bjørn Axel Security assessments of safety critical systems using HAZOPs [Book Section] // SafeComp'01 Proceedings of the 20th International Conference on Computer Safety, Reliability and Security / ed. Springer-Verlag. - London : [s.n.], 2001. - ISBN:3-540-42607-8.

Woskowski Christoph A Pragmatic Approach towards Safe and Secure Medical Device Integration [Conference] // 33rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP) / ed. Bondavalli Andrea and Giandomenico Felicita Di. - Florence : Springer International Publishing, 2014. - Vol. LNCS 8666. - pp. 342-353. - DOI: 10.1007/978-3-319-10506-2_23.

Yang Lili and Yang S. H. A Framework of Security and Safety Checking for Internet-Based Control Systems [Journal] // Int. J. Information and Computer Security. - 2007. - 1/2 : Vol. 1. - pp. 185-200.

Young William and Leveson Nancy G. Inside Risks: An Integrated Approach to Safety and Security Based on Systems Theory [Article] // Communications of the ACM. - 2014. - 2 : Vol. 57. - pp. 31-35. - DOI: 10.1145/2556938.

Zafar Saad and Dromey R. G. Integrating Safety and Security Requirements into Design of an Embedded System [Conference] // 12th Asia-Pacific Software Engineering Conference (APSEC). - [s.l.] : IEEE, 2005. - DOI: 10.1109/APSEC.2005.75.

8 Acronyms

Term/ abbreviation	Explanation
A	Action
AADL	Architecture Analysis and Design Language
ADO	Delivery and Operation
AFRL	Air Force Research Laboratory
AGD	Guidance Documents
AIP	Ambulatory Infusion Pump
ALARP	As Low As Reasonably Practicable
AMC	Acceptable Means of Compliance
ANS	Air Navigation Service
ARE	Admiralty Research Establishment
ARM	ARgument Meta-model
AT	Attack Tree
AVA	Vulnerability Assessment
AVATAR	Automated Verification of reAl Time softwARe
BACS	Building Automation and Control System
BBN	Bayesian Belief Network
BDMP	Boolean logic Driven Markov Process
BEV	Battery-Electric Vehicles
CAA	Civil Aviation Authority
CAE	Claims-Argument-Evidence
CBT	Component Behaviour Tree
CC	Common Criteria
CENELEC	Committee for Electro-technical Standardization
CFT	Component Fault Tree
CHASSIS	Combined Harm Assessment for Safety and Security of Information Systems
CHAZOP	Control HAZard and OPerability
CIA	Confidentiality, Integrity and Availability
CIN	Component Interaction Network
CLM	Component Logic Model
CLUSIF	French Cyber-Security Club (In French: Club de la Sécurité de l'Information Français)
CMMI	Capability Maturity Model Integration
CObIT	Control Objectives for Information and related Technology
CPS	Cyber-Physical System
CR	Critical Resource

CS	Computer System
CS	Certification Specifications
CSP	Communication Sequential Process
CWE	Common Weakness Enumeration
D	Defence
DAH	Design Approval Holder
DAL	Development Assurance Level
DAR	Decision, Analysis and Resolution
DARPA	Defense Advanced Research Project Agency
DDQS	Design, Develop and Qualify the Solution
D-MUC	Diagrammatical Misuse Case
DoD	Department of Defence
DoS	Denial of Service
DRA	Defence Research Agency
E/E/PE	Electrical / Electronic / Programmable Electronic
EAL	Evaluation Assurance Level
EASA	European Aviation Safety Agency
EFT	Extended Fault Tree
ERTMS	European Railway Traffic Management System
ETCS	European Train Control System
ETSI	European Telecommunications Standards Institute
EWICS	European Workshop on Industrial Computer Systems Reliability, Safety and Security
F	Failure
FA	Free Agent
FAA	Federal Aviation Authority
FAIR	Factor Analysis of Information Risk
FAR	Federal Aviation Regulation
FCV	Fuel-Cell Vehicle
FDA	Food and Drug Administration
FHA	Functional Hazard Analysis
FMEA	Failure Mode and Effect Analysis
FPTC	Failure Propagation and Transformation Calculus
FSD	Failure Sequence Diagram
FT	Fault Tree
GEMS	Generic Error-Modelling System
GM	Guidance Material
GPP	General Purpose Processor
GSE	Genetic Software Engineering

GSN	Goal Structuring Notation
GWV	Greve, Wilding, and Vanfleet
H	Harm
HACMS	High-Assurance Cyber Military Systems
HazOp	HAZard and Operability
HEV	Hybrid Electric Vehicles
HLR	High-Level Requirement
I&C	Instrumentation and Control
IACS	Industrial Automation and Control System
IAEA	International Atomic Energy Agency
IBT	Integrated Behaviour Tree
ICAO	International Civil Aviation Organisation
iCMM	integrated Capability Maturity Model
ICS	Industrial Control System
IDS	Intrusion Detection System
IMA	Integrated Modular Avionics
INCOSE	International Council on System Engineering
ISR	Instruction Set Randomization
KUL	Katholieke Universiteit Leuven
LLR	Low-Level Requirement
LOPA	Layer-Of-Protection Analysis
LSP	Liskov Substitutability Principle
MAFTIA	Malicious-and Accidental-Fault Tolerance for Internet Applications
MBS&SA	Model Based Safety & Security Assessment
MCS	Machine-Control Systems
MCS	Minimal Cut Set
MILS	Multiple Independent Levels of Security (obsolete)
MLS	Multiple Levels of Security
MOD	Ministry of Defence (UK)
MSC	Minimal Sufficient Condition
MUSD	Misuse Sequence Diagram
NFR	Non-Functional Requirement
NIST	National Institute of Standards and Technology
NSA	National Security Agency
O	Operator
OE	Operational Environment
OMG	Object Management Group

ONERA	Office National d'Études et de Recherches Aérospatiales (The French Aerospace Lab)
OS	Operating System
OSI	Open Systems Interconnection
P	Probability
PP	Protection Profile
PSM	Practical Software and Systems Measurement
PSSA	Preliminary System Security Assessment
R	Rating
RAE	Requirements Analysis and Elicitation
RBT	Requirement Behaviour Tree
RESS	Rechargeable Energy Storage System
RFT	Request For Tender
RTCA	Radio Technical Commission for Aeronautics
S	Secret
SACM	Structured Assurance Case Meta-model
SAEM	Software Assurance Evidence Meta-model
SAL	Symbolic Analysis Laboratory
SAM	Safety Assessment Methodology
SAT	Satisfiability
SaTrAp	Safety Traceability Approach
SCA	Software Communication Architecture
SCIS	Software-intensive Critical Information Systems
SDR	Software Defined Radio
SeCM	Security Conceptual Model
SEFT	State/Event Fault Tree
SEISES	Secured and Safe IT Embedded Systems
SEMA	System vs. Environment & Malicious vs. Accidental
SIL	Safety Integrity Level
SIS	Safety Interlock System
SL	Security Level
SL	Single Level (of Security)
SMT	Satisfiability Modulo Theory
SQUALE	Security, Safety and Quality Evaluation for Dependable Systems
STAMP	System-Theoretic Accident Model and Processes
STUK	Radiation and Nuclear Safety Authority (of Finland)
TCS	Thales Communications & Security
TEPE	TEmporal Property Expression
T-MUC	Textual Misuse Cases
TOE	Target of Evaluation

TRT	Thales Research & Technology
TS	Top Secret
TSF	Target of Evaluation Security Function
TSFI	TSF Interface
TTOOL	TURTLE Tool
TVRA	Threat, Vulnerability and Risk Assessment
VIA	Vulnerability Identification and Analysis