Safecomp FastAbstract 25th September 2013

Marc-Olivier Killijian

Safecomp FastAbstract Program

25th September 2013 16:00-17:30

Each FastAbstract will have a 60 seconds short presentation on the 25th from 16:00. Then the posters will be presented until 17:30.

1 - C. Arar, H. Kalla, S. Kalla and B. S. Sabrina
Fault-Tolerant Real-Time Scheduling Algorithm for Energy-Aware Embedded Systems

2 - I. Silva and L. A. Guedes, P. Portugal and F. Vasques
Common Cause Failure Analysis for Wireless Sensor Networks

3 - T. Probst, E. Alata, M. Kaâniche, V. Nicomette, Y. Deswarte
An Approach for Security Evaluation and Analysis in Cloud Computing

4 - M. Machin, J.-P. Blanquart, J. Guiochet, D. Powell and H. Waeselynck
Specifying Safety Monitors for Autonomous Systems

5 - J. Dittmann, T. Hoppe, C. Vielhauer
Multimedia Systems as Immune System to Improve Automotive Security?

6 - S. Shida, A. Uchida, M. Ishii, M. Ide, and K. Kuramitsu
Assure-It: A Runtime Synchronization Tool of Assurance Cases

7 - K. Łukasiewicz, J. Górski
Integrating agile practices into critical software development

8 - A. Kumar
Outsourced Linear Algebra

9 - R. Bloomfield and R. Stroud
Security-informed Safety

10 - A. Ruiz, H. Espinoza, T. Kelly
Adequacy of Contract Grammars for Component Certification

11 - A. Zammali, A. de Bonneval, Y. Crouzet
Communication Integrity for Slow-dynamic Critical Embedded Systems

Fault-Tolerant Real-Time Scheduling Algorithm for Energy-Aware Embedded Systems

Chafik Arar, Hamoudi Kalla, Salim Kalla and Bendib Sonia Sabrina
Department of Computer Science, University of Batna, Algeria
Email: {arar.chafik,hamoudi.kalla}@gmail.com, {salim.kalla,Bendib.SS}@univ-batna.dz

***Abstract*—In this paper, we propose a fault-tolerant scheduling approach that achieves low energy consumption and high reliability efficiency. Our scheduling solution is dedicated to multi-bus heterogeneous architectures, which take as input a given system description and a given fault hypothesis. It is based on active redundancy to mask a fixed number k of failures supported in the system, so that there is no need for detecting and handling such failures. In order to maximize the system's reliability, the replicas of each operation are scheduled on different reliable processors. Our solution can maximize reliability and reduce energy consumption when using active redundancy.**

I. Introduction

Embedded systems invade many sectors of human activity, such as medical applications, transportation, energy production, robotics, and telecommunication. The progresses achieved in electronics and data processing improves the performances of these systems. As a result, the new systems are increasingly small and fast, but also more complex and critical, and thus more sensitive to faults. The presence of some faults in these systems, accidental (design, interaction, ...) as well as intentional (human, virus, ...), are inevitable. Due to catastrophic consequences (human, ecological, and/or financial disasters) that could involve a fault, these systems must be fault-tolerant [1]. A fault can affect either the hardware or the software of the system. In this paper, we consider only processors faults in distributed architectures with buses. A bus is a multipoint connection characterized by a physical medium that connects all the processors of the architecture. As we are targeting embedded systems with limited resources (for reasons of weight, encumbrance, energy consumption, or price constraints), we investigate only software solutions.

We address hardware fault-tolerant approaches based on scheduling algorithms, to tolerate processors faults in distributed architectures. The approach that we propose is our most recent work for building fault-tolerant distributed embedded real-time systems. Prior results have been published in [2], [3]. In this paper, we are interested in approaches based on scheduling algorithms that maximize reliability and reduce energy consumption when using active redundancy to tolerate processors faults.

The paper is organized as follows. Section II describes the system model and states the faults assumptions. Section III presents our approach for providing fault-tolerance. Finally, Section IV concludes the paper.

II. Models

The algorithm is modeled as a data-flow graph, called algorithm graph and noted ALG. Each vertex of ALG is an operation and each edge is a data-dependence. A data-dependence corresponds to a data transfer between a producer operation and a consumer operation. o1 $\rightarrow$ o2 means that o1 is a predecessor of o2, and o2 is a successor of o1. Figure 1(a) presents an example of an algorithm graph, with eight operations I, I',A, B, C, D, O and O'.

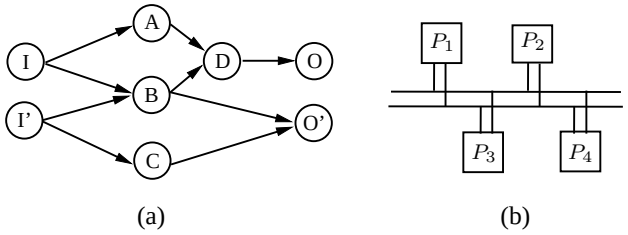


Fig. 1. Algorithm and architecture graphs.

The architecture is modeled by a non-directed graph, noted ARC, where each node is a processor, and each edge is a bus. We assume that the architecture is heterogeneous and fully connected. Figure 1(b) is an example of ARC, with four processors P_1, P_2, P_3 and P_4, and two buses Bus_1 and Bus_2.

We assume only hardware components failures and we assume that the algorithm is correct w.r.t. its specification, i.e., it has been formally validated, for instance with model checking and/or theorem proving tools. We consider only transient processors faults. We assume that at most k processors faults can arise in the system, and that the architecture includes more than k processors.

III. The proposed approach

The solution that we propose to tolerate processors faults is based on the active redundancy of operations. The advantage of the active redundancy is that the obtained schedule is static; in particular, there is no need for complex on-line re-scheduling of the operations that were executed on a processor when the latter fails; also, it can be proved that the schedule meets a required real-time constraint, both in the absence and in the presence of faults. In many embedded systems, this is mandatory.

To tolerate upto k arbitrary processors faults, each operation o of ALG is actively replicated on k+1 processors of ARC. For example, to tolerate two processors faults, three replicas of each operation of the ALG given in Figure 1(a) are scheduled on different processors (see Figure 3). We assume that all values returned by the k+1 replicas of any operation o of ALG are identical.

Figure 2 presents an example of a fault free schedule, where operations are not replicated.

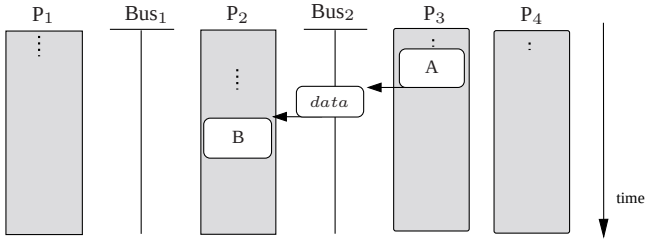


Fig. 2. A fault-free schedule.

In order to maximize the reliability R of the schedule, we propose to use the Global System Failure Rate per time unit (GSFR) function that we have proposed in [3]. The GSFR is the failure rate per time unit of the obtained multiprocessor schedule. In our approach, the replicas of each operation are scheduled on the best processors that minimizes GSFR.

The GSFR of scheduling an operation o_i, noted Λ, is computed by the following equation:

$$\Lambda(S_n) = -\log R(S_n)/U(S_n) \tag{1}$$

where, S_n is the static schedule at step n of the algorithm, and $U(S_n)$ is the total utilization of the processors.

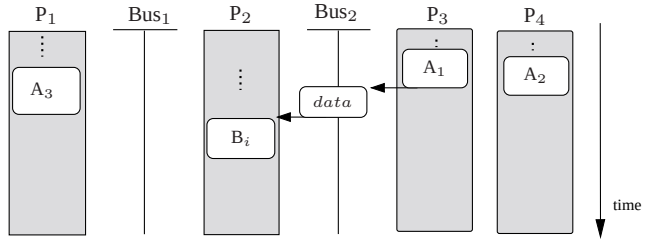


Fig. 3. A reliable fault tolerant schedule.

A. Voltage, frequency, and energy consumption

The maximum supply voltage is noted V_{max} and the corresponding highest operating frequency is noted f_{max}. For each operation, its execution time Exe assumes that the processor operates at f_{max} and V_{max}. The execution time of the operation placed onto the processor p running at frequency f (taken as a scaling factor) is:

$$Exe(X, p, f) = Exe(X, p)/f \tag{2}$$

Concerning the power consumption, we follow the model of Zhu et al. [4]. For a single operation placed onto a single processor, the power consumption Pc is:

$$Pc = P_s + h(P_{ind} + P_d) \tag{3}$$

where, $P_d = Cef\ V^2\ f$, Ps is the static power (power to maintain basic circuits and to keep the clock running), h is equal to 1 when the circuit is active and 0 when it is inactive, P_{ind} is the frequency independent active power, P_d is the frequency dependent active power, Cef is the switch capacitance, V is the supply voltage, and f is the operating frequency.

For a multiprocessor schedule S, we cannot apply directly equation 3. Instead, we must compute the total energy E(S) consumed by S, and then divide by the schedule length L(S):

$$P(S) = E(S)/L(S) \tag{4}$$

In our approach, as the k+1 replicas of each operation are scheduled actively on k+1 distinct processors, the energy consumed by the system is maximal. In order to reduce energy consumption, we propose to execute the k+1 replicas of an operation with different frequencies f. As all the k+1 replicas of each operation may have different end execution time (see Figure 3 for the replicas of A), we choose to align the execution time of all the replicas by changing the frequency f of each replica. For example, to reduce energy consumption of the system of Figure 3, we align the first two replicas A_1 and A_2 with A_3 by changing frequencies as shown in Figure 4.

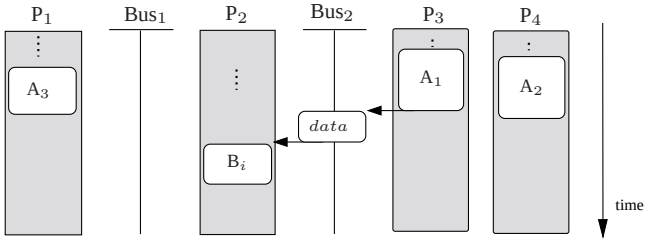


Fig. 4. The proposed scheme to reduce energy consumption

The new frequency of each replica o_i of o is depend on the end execution time of the last scheduled replica of o.

IV. Conclusion

We have proposed in this paper a solution to tolerate several processors faults in distributed heterogeneous architectures with multiple-bus topology. The proposed solution is based on active redundancy, and on GSFR to maximize reliability and to reduce energy consumption. Currently, we are working to evaluate our approach.

References

[1] N. Suri and K. Ramamritham, "Editorial: Special section on dependable real-time systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 10, no. 6, pp. 529–531, Jun. 1999.

[2] I. Assayad, A. Girault, and H. Kalla, "Tradeoff exploration between reliability, power consumption, and execution time for embedded systems - the tsh tricriteria scheduling heuristic," *STTT*, vol. 15, no. 3, pp. 229–245, 2013.

[3] A. Girault and H. Kalla, "A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate," *IEEE Trans. Dependable Sec. Comput.*, vol. 6, no. 4, pp. 241–254, 2009.

[4] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *International Conference on Computer Aided Design, ICCAD'04*, San Jose (CA), USA, Nov. 2004, pp. 35–40.

Common Cause Failure Analysis for Wireless Sensor Networks

Ivanovitch Silva and Luiz Affonso Guedes
Federal University of Rio Grande do Norte, Natal, Brazil
{ivan,affonso}@dca.ufrn.br

Paulo Portugal and Francisco Vasques
University of Porto, Porto, Portugal
{pportugal,vasques}@fe.up.pt

***Abstract*—Simultaneous failures of multiple devices make the dominant contribution to the unreliability of wireless sensor networks. They can hamper communications over long periods of time and consequently disturb, or even disable, the management algorithms of the network. In this preliminary work, we consider two types of common cause failures: hardware, and the temporary disruption of links. We propose an evaluation methodology based on the fault tree formalism for analyzing the reliability and availability of any wireless sensor network when common cause failures are considered.**

***Index Terms*—Common cause failure, fault tree, reliability, availability, wireless sensor network.**

I. Introduction

In wireless sensor networks (WSN), the failure of a single device might not be critical to the applications due to its intrinsic redundancy. However, when simultaneous failures occur with multiple devices, the consequences are likely to be disastrous, particulary for critical applications (patient surveillance, industrial environments, safety monitoring) [1]. This type of failure is known as common cause failure (CCF).

It is very important to measure the impact of CCF as soon as possible, ideally in the early phases of planning and designing the network. When done properly, such an early evaluation can anticipate decisions regarding the topology, criticality of the devices, the levels of redundancy, and network robustness. A tentative effort to evaluate a WSN considering CCF was performed in [2], but the technique was focused on a single cluster. By introducing the concept of coverage-oriented reliability, the same authors extended the previous work in [2] to support a more flexible way to configure failure conditions [3]. However, it is not possible to create two or more coverage subsets for the same cluster. In [4], the effects of CCF on a WSN were also evaluated. The authors proposed a progressive scheme based on binary decision diagrams for evaluating any WSN topology. However, the model supports neither generic network failure conditions nor an importance analysis of the devices.

The main focus here is to investigate the influence of common cause failures on a WSN. It becomes clear from the above discussion that previous research has only provided a partial solution for this problem. We proposed a methodology based on the fault tree formalism in [5] to evaluate the reliability and availability of a WSN, supporting the definition of flexible network failure conditions. Here, we extend this methodology to support CCF for both hardware and links.

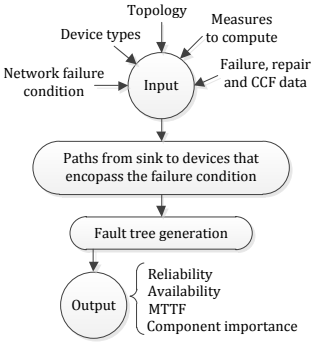


Fig. 1. Overview of the methodology for common cause failure analysis.

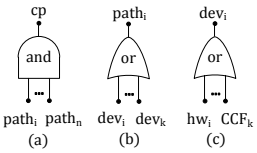


Fig. 2. Device failure condition.

II. Evaluation Methodology Incorporating Common Cause Failures

An overview of the methodology adopted in this preliminary work is given in Fig. 1. The process starts by providing information about the network topology, device types, device failure and repair processes, CCF data, and network failure condition. The latter expresses the conditions that may lead to a network (system) failure, and is defined by a logical expression that combines the failure status of the devices. The next step is to find all the paths between the sink and devices that encompass the failure condition. This is necessary for attaining flexible failure conditions and for supporting self-healing routing protocols. In the following step, a fault tree is generated using all the previous data. Finally, the metrics of interest (reliability, availability, MTTF, and importance measures) are computed.

A. Assumptions

The main assumptions of this methodology are summarized as follows:

- **Faults:** only permanent faults are considered. The links, due their wireless nature, are more affected by transient faults (millisecond scale). However, temporary barriers or bad weather conditions can obstruct a link for long hours. This case is also considered as a permanent fault. Failure occurrences are characterized by a *failure distribution*, by means of a CDF (cumulative distribution function). Any type of CDF can be used to describe their occurrence.

- **CCF:** the model support two types of CCF: hardware and link. The former is caused by shocks or other actions that damage the device hardware, whereas the latter is caused by temporary interruptions of a link (occurrence $\gg$ milliseconds). Depending on the scenario, the CCF can be fatal or non-fatal to the network.
- **Network failure condition:** the network failure condition (NFC) defines which combination of devices may lead to a network failure and its equivalent to the TOP event of fault tree. The methodology used in this letter supports any combination that can be expressed using boolean operators (i.e., AND, OR, K-*out-of*-N).

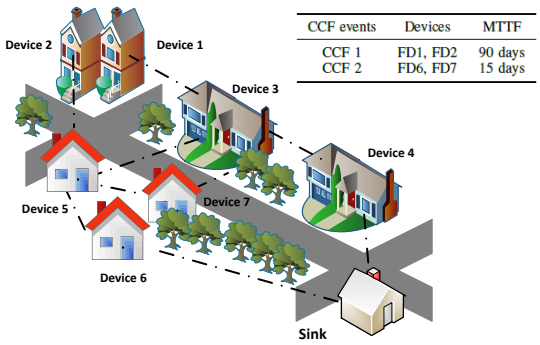
B. Device Failure Condition

After defining the NFC, it is necessary to define the conditions that may lead to the failure of a device. According to Fig. 2a, a device is considered faulty if all the paths between it and the sink fail (event cp – conectivity problem). On the other hand, as described in Fig. 2b, a path fails if at least one device along that path fails. Finally, a device also can fail if its hardware fails (event hw) or if at least one CCF occurs (Fig. 2c).

Note that it is necessary to exert some effort to find all combinations that may lead to a connectivity failure. In order to attain this, it is necessary to search all paths between the sink and devices that belong to the NFC. The paths are found by performing a depth-first search (DFS) in an adjacency matrix that represents the network. All the paths generated are then stored in a data structure based on a fault tree.

III. Preliminary Results

In this section, we evaluate the common cause failures for wireless sensor networks. The main target of the analysis is to highlight the influence of common cause failures upon network reliability. We assume a wireless sensor network mesh topology typical for condominium monitoring applications to validate the idea (Fig. 3). In this scenario, we consider that the network fails if at least three devices fail.



CCF events	Devices	MTTF
CCF 1	FD1, FD2	90 days
CCF 2	FD6, FD7	15 days

Fig. 3. Topology adopted in the evaluation process.

Regarding the failure properties, we assume that device failures occur at a constant rate. In order to simplify the procedure here, we assume that all devices have the same failure rate ($\lambda \approx 1E-5$).

The evaluation measures the influence of CCF on the network reliability. The results are described in Fig. 4. Despite the events CCF 1 and CCF 2 having different configurations, when both events occur the influence on network reliability is similar. This behavior results from the difference in criticality of devices 1, 2, and 7. Note that the network reliability decreases quickly when the events CCF 1 and CC2 are designed together. This scenario can even be pessimistic, but when compared to with the scenario without CCF, we observe that no consideration of CCF for network reliability is a very unrealistic assumption.

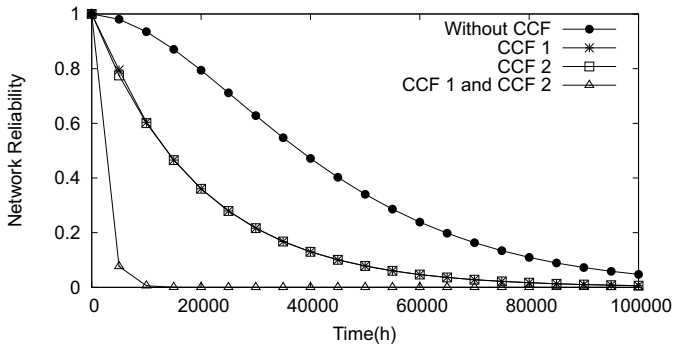


Fig. 4. Network reliability analysis considering common cause failure.

IV. Discussion and Future Work

In this preliminary work, we proposed an evaluation model for wireless sensor networks considering common cause failures. The proposal is based on the fault tree formalism and it considers hardware and link failures. A mesh topology was used for its validation (network reliability). It is also possible to analyze the network availability, and the criticality of the devices. The result showed the importance of considering CCFs during the design of a network, mainly when multiples CCF can occur. This analysis can be used to design any wireless sensor network. In future research, we plan to extend this methodology to analyze the redundancy, coverage factor, and hierarchical models.

References

[1] R. Dilmaghani, H. Bobarshad, M. Ghavami, S. Choobkar, and C. Wolfe, "Wireless sensor networks for monitoring physiological signals of multiple patients," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 347–356, 2011.

[2] A. Shrestha, L. Xing, and H. Liu, "Infrastructure communication reliability of wireless sensor networks," in *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*, 2006, pp. 250 –257.

[3] A. Shrestha, H. Liu, and L. Xing, "Modeling and evaluating the reliability of wireless sensor networks," in *Reliability and Maintainability Symposium, 2007. RAMS '07. Annual*, jan. 2007, pp. 186 –191.

[4] L. Xing, H. Liu, and A. Shrestha, "Infrastructure communication reliability of wireless sensor networks considering common-cause failures," *International Journal of Performability Engineering*, vol. 8, no. 2, pp. 141–150, 2012.

[5] I. Silva, L. A. Guedes, P. Portugal, and F. Vasques, "Reliability and availability evaluation of wireless sensor networks for industrial applications," *Sensors*, vol. 12, no. 1, pp. 806–838, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/1/806/

An Approach for Security Evaluation and Analysis in Cloud Computing

T. Probst[1,2], E. Alata[1,3], M. Kaâniche[1,4], V. Nicomette[1,3], Y. Deswarte[1,4]
[1]CNRS, LAAS, 7 Avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, INP de Toulouse, LAAS F-31400 Toulouse, France
[3]Univ de Toulouse, INSA de Toulouse, LAAS F-31400 Toulouse, France
[4]Univ de Toulouse, LAAS, LAAS F-31400 Toulouse, France
{probst,ealata,kaaniche,nicomett,deswarte}@laas.fr

***Abstract*—This paper describes a novel approach for security evaluation and analysis in cloud computing environments. The objective is to provide an automated way to evaluate the efficiency of security mechanisms aiming at protecting the cloud computing infrastructures and applications. In particular, we focus on access controls and intrusion detection/prevention systems. We leverage cloud benefits to optimize the audit and assessment processes. The proposed approach is currently under implementation on our cloud platform.**

I. Introduction

By offering various service and deployment models [1], cloud computing provides easy ways to host and manage infrastructures and applications, while reducing deployment and operation costs. However, the rapid development of cloud computing over the last few years has raised new security concerns [2], [3]. Enforcing the security and dependability of cloud computing infrastructures is necessary to allow their use for the deployment of critical applications. Different levels of protection are needed to implement security policies, as done in traditional environments. Nevertheless, there are specific characteristics and challenges that need to be taken into account carefully in cloud computing environments. Among others, we can cite: 1) the co-residence of several clients on the same physical infrastructure; 2) the mix of different technologies like virtualization, new network architectures, Web applications and services; 3) the co-existence of different levels of security controls on the client side and on the cloud provider side; and 4) the emergence of new attacks involving cloud infrastructures. Thus, usual security mechanisms must be adapted so they can be efficient in such a context. Our purpose it to assess the efficiency of these mechanisms.

Except for recommandations such as the Cloud Security Alliance guidance on security assessments [4], very few approaches for cloud security evaluation have been developed so far. We can cite Bleikertz's work [5], addressing security audits in public cloud infrastructures. His approach allows to generate attack graphs from network accessibility graphs and vulnerability scans. However, it only addresses network-based attacks and the attack scenarios found are never executed. Furthermore, no in-line firewall or Intrusion Detection/Prevention Systems (IDS/IPS) evaluation is performed.

In this paper, we propose an approach to efficiently conduct automated security evaluations and analysis of Infrastructure as a Service cloud computing environments. The ultimate goal is to give the client a detailed picture of the risks he takes by using the cloud, and the provider a good insight of what sort of threats a client may represent. In particular, we are interested in two challenges regarding cloud computing security: access controls (including network and user access filtering) and intrusion detection and prevention. To evaluate the efficiency of these security measures, we follow a two-phase process: 1) access control analysis to evaluate the performance of network and user access filtering, and get the accessibilities; 2) IDS/IPS evaluation by executing relevant attack scenarios that take into account the accessibilities found.

Further details about the proposed approach are presented in the remainder of this paper.

II. Proposed approach

Various security mechanisms are deployed in the cloud to implement security policies: firewalls, IDS/IPS, Identity and Access Management (IAM) tools. Our objective is to verify that they are correctly deployed and configured to fit the security requirements of the client and of the provider.

Our methodology and its corresponding steps are illustrated in Fig. 1. It is based on the elaboration and execution of attack scenarios targeting the hosts involved in the client's virtual infrastructure. As these attacks may compromise hosts and, as a consequence, could interfere with the client's business, we chose to clone (step a) this infrastructure (virtual datacenters: networks, firewalls, machines and applications) and perform attacks on this clone. Furthermore, cloning the client's infrastructure allows us to purposely inject vulnerabilities on the different hosts, to set up and perform complex attack scenarios, which is particularly useful when one needs to assess the efficiency of IDS/IPS in the presence of some specific vulnerabilities and attacks. We take into account the presence of other potentially malicious concurrent clients that we represent through other infrastructures we control.

To be automated, our approach takes advantage of cloud computing embedded technologies to run audit operations (cloning infrastructures, executing virtual machines, deploying applications).

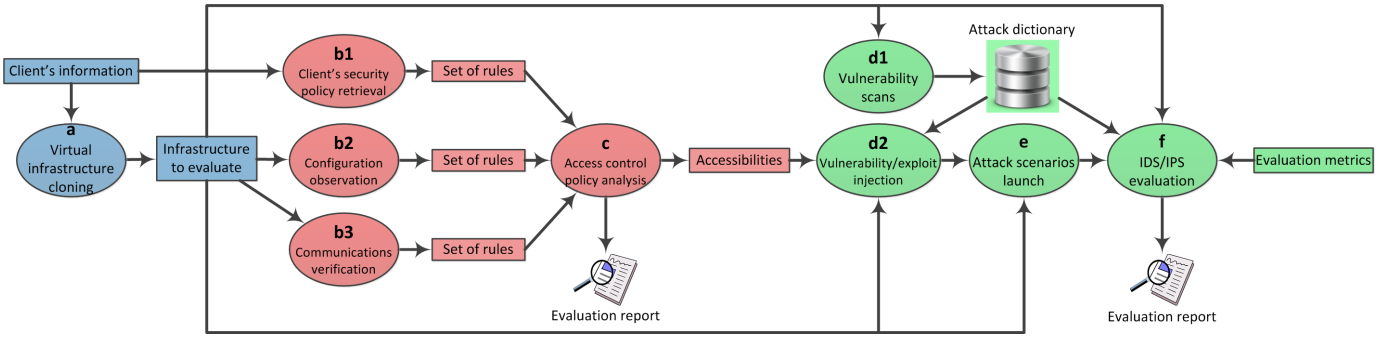


Fig. 1. Overall evaluation and analysis process

A. Analysis of access control policy

The list of authorized communications (so-called accessibilities) inside the infrastructure to be assessed, is identified using three ways:

1) By retrieving statically the client's security policy, written in a formal or informal specification document (step b1).
2) By extracting information from the cloned cloud component's configuration (networks, hypervisor-based firewalls, virtual firewalls, IAM tools) (step b2).
3) By performing experiments: sending traffic such as network sweeps in order to verify the effective possible communications (step b3).

These accessibilities are translated into a set of handy rules. We chose the Prolog logic programming language because it is well adapted to express predicates and compute some logic to deduce rules characterizing the accessibilities, such as:

```
communication(source, sproto, sport, destination, dproto, dport).
```

This rule specifies the kind of traffic a source object (IP address, network, security group...) can send to a destination object. By processing these rules, one can identify inconsistencies in the implementation of the policy (normally the three set of rules should be equivalent), and deduce the accessibility matrix (taking the second or the third set of rules). The confrontation of the outcomes highlighting the inconsistencies produces an access control policy evaluation as a first result (step c).

B. Evaluation of Intrusion Detection and Prevention Systems

To compose attack scenarios, we need to use a simple attack taxonomy based on the attack vector. For this purpose, we follow a dimension-based classification [6], using the attack vector as the dimension. To be as exhaustive as possible in the evaluation of the IDS/IPS, we plan to execute every possible attack category on behalf of every possible attacker (insiders and outsiders) and towards every possible target. The actually running virtual machines and their images are first scanned to find potential vulnerabilities (step d1). Additional exploitable vulnerabilities that are associated to the accessibilities found (step d2) will be injected on purpose to check the IDS/IPS reaction under some specific attack campaigns for which alarms are expected to be raised. A dictionary containing the description of attacks (exploit, vulnerability, environment parameters, and expected results) for each class of our taxonomy is used to perform the injection of these vulnerabilities and their matching exploits. The preconditions, postconditions and objectives are deduced from the attack scenarios built and launched on the fly (step e). The output of the Security Information and Event Management (SIEM), which aggregates and correlates the IDS/IPS probes, along with the state of the targets, will be used to give us the verdict to know whether the attacks have been detected/prevented or not, according to the expected results from the aforementioned dictionary (step f). Finally, quantitative metrics will be derived to assess false negative and false positive detection rates for each attack class, and also to identify optimal configurations of the IDS/IPS in the cloud infrastructure.

III. Conclusion

This paper presents the principles of an approach we are developing to automatically conduct security evaluations and analysis in cloud computing infrastructures. Our approach is aimed at optimizing the overall process while providing accurate assessments of the major security aspects of the cloud. The proposed approach is currently investigated and implemented using a VMware vCloud Suite platform including various security mechanisms.

Acknowledgment

This research is supported by the French project Investissements d'Avenir Secured Virtual Cloud (SVC).

References

[1] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v3.0", 2011.
[2] M. Jensen et al., "On Technical Security Issues in Cloud Computing", in *IEEE International Conference on Cloud Computing*, Bangalore, India, 2009, pp. 109-116.
[3] I. Studnia et al., "Survey of Security Problems in Cloud Computing Virtual Machines", in *Computer and Electronics Security Applications Rendez-vous*, Rennes, France, 2012, pp. 61-74.
[4] Cloud Security Alliance, "SecaaS Implementation Guidance: Security Assessments", 2012.
[5] S. Bleikertz, "Automated Security Analysis of Infrastructure Clouds", M.S. thesis, Department of Informatics and Mathmatical Modelling, Technical University of Denmark, and Department of Telematics, Norwegian University of Science and Technology, 2010.
[6] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks", in *Computers and Security*, 2005, 24 pp. 31-43.

Specifying safety monitors for autonomous systems

Mathilde Machin*†, Jean-Paul Blanquart‡, Jérémie Guiochet*†, David Powell*† and Hélène Waeselynck*†

* *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
† *Univ de Toulouse, LAAS, F-31400 Toulouse, France*
{mmachin | guiochet | dpowell | waeselynck}@laas.fr

‡ *EADS Astrium, 31 rue des cosmonautes, 31402 Toulouse, France*
jean-paul.blanquart@astrium.eads.net

I. Introduction

Autonomous systems aim to be versatile and able to perform tasks in various ill-defined environments. These systems are often critical since their failure can lead to large financial losses or human injury. In this context, classical safety measures are inflexible and are not sufficient to guarantee that the system behaves safely. For instance, emergency stop buttons, if used alone, transfer responsibility for safety to the user, which is clearly inadequate for a system that is supposed to be autonomous. Electromechanical solutions such as bumper motor switches reduce versatility since, for example, they cannot be used for systems that need to push objects or operate in the presence of fragile obstacles.

As a result, autonomous systems have to be equipped with means for context-dependent safety enforcement. We consider here a device called a safety monitor, which is equipped with the necessary means for context observation (i.e., sensors) and able to trigger, when necessary, the appropriate safety action. We require the monitor to be *maximally permissive*, in that it should only restrict its *versatility* (i.e., what the system is able to do) to the extent necessary to ensure safety.

Flexible safety measures are used in the robotic domain in [1], where the safety context is simply the distance from the robot to a human, because only the robot-to-human collision hazard is taken into account. A richer context is needed in order to cover a wider range of hazards, obtained through hazard analysis, as in [2].

Once a hazard is identified by hazard analysis, it is necessary to specify what the monitor has to do to avoid it. This is not straightforward. To determine when to act, precursory conditions have to be extracted from the hazard analysis. Obviously, how to intervene is also of interest and according to the chosen strategy, the precursory conditions may be different. We call the couple when/how or observation/intervention a *safety rule*, which is a requirement for the safety monitor.

We distinguish *initiative* and *restriction* rules. An initiative rule launches an *action* in order to change the state. On the contrary a restriction rule *inhibits* certain state changes, e.g., by means of an interlock device or by request filtering.

This work is partially supported by the SAPHARI Project, funded under the 7th Framework Programme of the European Community.

Synthesis of restriction rules has been widely studied in the context of *supervisory control* [3], which is a formal method that generates a 'controller' from a system model and a constraint model (to avoid the catastrophic state in our case), both expressed as automata. In this context, 'control' means forbidding the occurrence of certain controllable events.

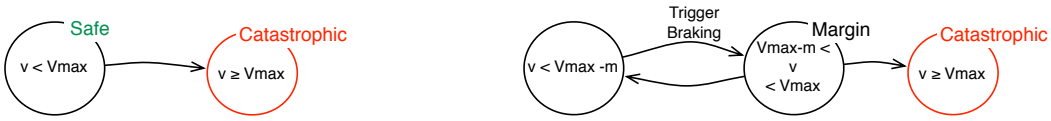
However, the need for initiative rules arises when inhibitions alone cannot ensure safety or are inefficient. For instance, to avoid a mobile obstacle, the triggering of 'brake' or 'swerve' actions would be more efficient than forbidding 'acceleration'. Since any such action takes a non-zero time to produce an effect (for example, braking in order to stop), it has to be anticipated by some *safety margin* (with respect to time or some other physical variable)

Our approach for specifying safety monitors is based on hazard analysis and considers both initiative and restriction rules. After a brief summary of previous work, we give the directions of our current research aimed at strengthening requirement elicitation by the use of formal methods.

II. Previous work

Previous work [4] has addressed the process for eliciting safety rules, based on a HAZOP-UML hazard analysis [5]. The system is described abstractly with UML use cases and sequence diagrams. Each row of a HAZOP table considers a deviation of the UML model and its consequences, assigning a severity level to it. For each deviation with a severity level of "serious" or higher, a constraint is formulated in natural language, by negation of the deviation or of an effect of the deviation.

The constraint is then expressed formally as an invariant, with predicates on variables that are observable by the monitor. A region graph is built from the invariant; the region violating the invariant is called the catastrophic region. Each transition leading to the catastrophic region has to be neutralized either by an inhibition or by insertion of an action and an associated safety margin (e.g., see Figure **??**). Safety rules elicited from each deviation should be applied in the context of the considered use case. The currently applicable use case is assumed to be identified on-line by the safety monitor. The problem of simultaneous and potentially conflicting interventions is addressed by composition of graphs and manual analysis.



(a) Region graph from invariant

(b) Region graph with a margin state

Fig. 1: Example of margin insertion in region graph applied to a simple speed limitation

III. Current work

We are currently extending the three steps of the work described in Section II: hazard analysis, safety modeling and safety rule elicitation.

1) HAZOP-UML improvements: Since a use case may in fact encompass several different situations, different safety rules can be required. We need contexts at a finer granularity than use cases. We extend the HAZOP table by the addition of a contextual information column. It aims to identify in the whole context (speed, temperature, human distance...) the relevant conditions under which the considered deviation leads to a given consequence. This extra piece of information enables the safety rules to be applied only in relevant cases and incites experts to discover other contexts where the consequences are different.

2) Towards a formal model: HAZOP-UML is an informal analysis using natural language whereas we aim to have a computable model. To ease formal modeling, we propose to disambiguate the constraint formulated from a HAZOP row by using a CNL (Controlled Natural Language), which is a natural language with restriction on syntax and vocabulary [6].

In addition, we aim to propose a template for analyzing each constraint. Safety-relevant variables are identified as well as thresholds. Unobservable conditions have to be replaced by predicates on observable variables. We also need to reconsider the differences between the ideal physical variables mentioned in HAZOP and their logical representation within the monitor, taking account of sensor accuracy, precision, range, sampling rate, and so on. Once the region graph is built, potential actions are examined. If the observed variables are controllable, this is quite straightforward. In case of uncontrollable variables, indirect actions have to be considered, which could imply additional observable variables. Actions require the insertion of margin regions. Variables, thresholds and actions are iteratively added to the graph until it models the state space behavior of the complete set of safety rules.

3) Formal support: We aim to improve the scalability of the method by the use of formal methods applied at each of three steps:

a) *Design and checking of safety rules for one constraint.* Model-checking can be used to guarantee that the local catastrophic state is inaccessible. Supervisor synthesis can find possible restrictions. It may not be possible to define an automatic method to find initiative rules. However, initiatives might be suggested from a catalogue of actions, with models of their effects on observable variables.

b) *Evaluation of redundancy between constraints.* We have found from several case studies that it is quite common for a single safety rule to cover constraints from several HAZOP rows. To discover this automatically, we need a formal model, i.e., the safety invariant, and a formal method that can reason about predicates on real variables, such as an SMT (Satisfiability Modulo Theory) solver.

c) *Validation of the overall set of rules.* Once all safety rules are defined, all relevant thresholds are identified and real domains with their predicates are reduced to enumerated sets of values. Therefore, we could use classical boolean model-checkers, to check overall properties such as the presence of simultaneous actions. To assess monitor permissiveness, versatility has to be modeled either by CTL (Computational Tree Logic) properties or by graph models.

IV. Conclusion

Our current research aims to ensure correctness, completeness and consistency of safety monitor requirements by formal methods. The results of formal methods are exact. However, if the underlying models are inaccurate, exact results have no interest. Therefore, the modeling step has to be done very carefully (in our case, the constraint analysis). Using different formal methods, the problem of translation and equivalence between formalisms will arise. The underlying goal is to have a formal proof of correctness from hazard analysis to implementation.

Our method will be applied on an industrial robot that helps aeronautics workers to install brackets in aircraft. The task is a collaborative one: the robot projects an image and prepares the surface with a solvent, while the worker glues the bracket.

References

[1] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger, "Towards the robotic co-worker," in *Robotics Research*. Springer, 2011, pp. 261–282.

[2] R. Woodman, A. F. Winfield, C. Harper, and M. Fraser, "Building safer robots: Safety driven control," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1603–1626, 2012.

[3] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[4] A. Mekki-Mokhtar, J.-P. Blanquart, J. Guiochet, D. Powell, and M. Roy, "Safety trigger conditions for critical autonomous systems," in *18th. Pacific Rim Int'l Symp. on Dependable Computing (PRDC)*. IEEE, 2012, pp. 61–69.

[5] J. Guiochet, D. Martin-Guillerez, and D. Powell, "Experience with model-based user-centered risk assessment for service robots," in *12th Int'l Symp. on High-Assurance Systems Engineering (HASE)*. IEEE, 2010, pp. 104–113.

[6] R. Schwitter, "Controlled natural languages for knowledge representation," in *23rd Int'l Conf. on Computational Linguistics (COLING '10): Posters*. Association for Computational Linguistics, 2010, pp. 1113–1121.

Multimedia Systems as Immune System to Improve Automotive Security?

Jana Dittmann[1], Tobias Hoppe[1]
[1] Otto von Guericke University Magdeburg
Germany

Claus Vielhauer[1, 2]
[2]Brandenburg University of Applied Sciences
Germany

Abstract—**Our motivation is driven by the fact, that security mechanisms often cause additional efforts and costs, and need to be aligned with safety goals - protecting human and environment. Especially in the field of automotive security, producers are seeking cost efficient, environmental-condition-adaptive (robust) and fast approaches, if possible combined with existing concepts reusing resources. Initially, working in automotive security, it was easy to see that a wide variety of attacks is possible, e.g. using knowledge from classical computer and network security incidents. It became clear, that malicious activities on car IT systems might also lead to safety relevant issues such as accidents with threats to life or physical condition of the driver herself, occupants and people in the environment. We are inspired by the basic law of robotics established by I. Asimov[1] and apply it to automotive design: "1. A vehicle may not injure a human being, or, through inaction, allow a human being to come to harm. 2. A vehicle must obey orders given it by human beings except where such orders would conflict with the First Law. 3. A vehicle must protect its own existence as long as such protection does not conflict with the First or Second Law.". This entire bunch of requirements would cause "heavy" technology and expensive solutions. Coming from multimedia security, it became stepwise clearer that we cannot neglect known security mechanisms, but it is worth to combine them with knowledge about multimedia systems (MM systems). For example MM sensory, data streams, protocols, quality etc. enable the car to perceive the occupants and environment more precisely and can help to detect in-car and outside-car anomalies caused by security incidents with an estimation/prediction of harm. Existing automotive components designed for safety, entertainment and comfort services should be able to help in achieving secure and safe car behavior. In the article we discuss the opportunity to understand vehicles as single and cooperative MM systems with the ability to become an enabler for future automobile security detection, warning and reaction strategies – as a kind of vehicular immune system building Multimedia-enabled Asimovian Secure Automobiles (MASA).**

I. Multimedia Automobiles and Automobile Security challenges

IT has arrived in the automotive world since years. Comfort and safety goals guided and still guide the developments and car IT becomes increasingly complex; autonomous and self-interacting cars offer a wide variety of sensory and complex analysis logic for interaction and presentation.

The bad thing - Due to complex functional and structural component dependencies, vulnerabilities in design, implementation or configuration are very likely and may be exploited maliciously to cause security incidents. Motivations are manifold ranging from tuning, stealing, manipulations or unlocking of functions/restrictions up to infecting cars with malware disturbing or harming driver and occupants. Managing automotive security incidents has to face the challenge, that incidents should not cause any threat to life or physical conditions by ensuring fundamental Asimov laws, which is very seldom explicitly considered yet. Therefore several constraints arise, e.g. existing security mechanisms which stop and reconfigure or update hard- or software should not be applied whilst driving to avoid difficult handling by the driver. When vulnerabilities, e.g. as reported in [1], [2] and [3], get exploited in complex scenarios like downhill driving on a winding road, the car needs to recognize this as difficult situation and well-selected reactions are needed to keep the car well on the road. Using the findings from [4], we can summarize: the car needs complete and correct information of the initial state of the world, the car is the sole cause of change, and action execution is atomic, indivisible, and results in effects which are deterministic and completely predictable. Of course, the automotive world is not static or has complete information to derive and determine a perfect security mechanisms avoiding harm. The awareness of security as relevant business factor is raising, but producers need to balance their costs and resources by reusing existing components and implementations and try to reduce restrictions coming along with the introduction of security mechanisms.

II. The future: Multimedia enabled Security Incident Detection and Reaction

The good thing - automotive systems are mobile MM systems offering a wide variety of sensor and media technology enabling cars to perceive humans, other subjects and objects behavior or its environment and context more precisely. For example, car sensors determine inside climate, outside weather, road conditions, know vehicle speed even those from other objects, windows and doors states, distance to surrounding subjects and objects, can determine seat occupations, number of persons in the car, profile drivers and occupants (persons gender, age, weight, height) etc. This involves indeed a lot of MM acquisition aspects. Sensory data is communicated to electronic control units for analysis and appropriate adjustment, to driver – car-to-human – or, in future, to other cars (car-to-car) and infrastructure (car-to-infrastructure), e.g. to enable autonomous driving. Based on individual car sensors and available MM capabilities it should be possible to design and build a model about what a car can "see" and which information can be used and communicated for incident detection, warning and reaction. Based on safety functions in case of an incidents, the in-

[1] I. Asimov. 1942. Runaround. Astounding Science Fiction.

volved MM system should be able to determine the current driving situation and environment and which potential harm needs to be avoided. Further, it should be capable of a wide variety of driver support for handling incidents in a safe manner. Knowledge from MM systems can help to **better understand** car behavior, identify anomalies and implausible states. Of course car MM systems are not yet designed as security measure and there is still uncertainty about interplay of individual technical system states to the global system behavior on roads (e.g. imperfect data propagation). Conditions of individual drivers and occupants (health etc.), environment, other cars and car occupants etc. are very dynamic and not yet considered in a global manner and transferred into a context model to determine secure and safe states. E.g., [5] motivates complex multi-agent environment solutions as *adjustable autonomy* considering different humans' interaction with varying preferences, inaccuracies and uncertainties. In [6] vehicles behavior is understood as team context actions with uncertainty and mutual information mismatch. **How may MM systems become a use case for automotive security?** Some examples:

MM systems can support security incident handling to **make an informed decision by providing a detailed cross media analysis** of available data from car sensory and connected systems. Existing cross media analysis can help to further investigate fast algorithms in high speed drives and on highly dynamic media – with sensory data of different spatial/temporal resolution and quality, or with varying amounts of surrounding activities – e.g. activity level during low/high traffic.

Drivers and occupants interact with modern cars, environment and with infrastructure (other cars, their drivers etc.) causing complex interplay, further uncertainty, error or loss of information. **MM data needs to be interpreted in its context to determine secure and insecure states, potential error and loss within its dynamic environment**. A car context model might help to interpret cross media analysis results, considering different drives such as high/low speed or parking, to be able to predict harm and to achieve a more efficient reaction. **Enhancing context-based cross media analysis with consistency and plausibility checks** on signal, data, feature and application level are further examples. After malicious actions (such as handbrake activation requests at high driving speed), implausible behavior detected early in the particular context (analyzing all sensor data and fusing into driving context) can help to stop hazards already before they occur.

"*First aid*" *guidelines* can be defined and applied, controlling how the car should behave avoiding blackout situations before full recovery is performed in a secure and safe manner – e.g. in the next service station. Similar strategies are already applied today on failures of safety-critical components: default / fallback strategies like "limp home mode" for engine control units, override functions for automatic steering or emergency modes in the ESP system. For security incidents, a more fine granular analysis of component behavior and context seems possible to define sensor-dependent command variables enforcing normal behavior. Approaches might use **re-injection of all kind of available plausible data** known from context models to components/networks. Resources of other nodes could be shared via the network to take over functionality from a failing node, or considering **organic computing principles within MM context for self-healing**.

How to solve conflicts? To avoid harm, cross media analysis and car MM context can check consistency and plausibility to find anomalies and define reactions for the current driving context. Further questions arise: how to react when not all multilateral security and safety interests of the affected car(s) and human(s) can be satisfied? Is the safety of the infected car's driver more important than the safety of other humans – or how can the MM system help to understand cross-party content/context and communicate risks to others?

How can MM user interfaces be used to communicate security incidents and safety risks to drivers, occupants or even the environment? **Security warnings might be designed MM based and context-adaptive** (e.g. regarding speed and traffic), as already discussed e.g. for incident reactions [1]. Also the resilience behavior of individual drivers and occupants to technical incidents could be investigated and how MM systems can address this. Which MM elements (visual, acoustic, haptic …) should be selected, what advice should be given? After accidents, systems could evaluate context criteria to select first aiders and derive/communicate appropriate instructions [7].

In summary, it is worth to extensively study and enhance MM technology means as vehicular immune system to enhance security incident detection, reaction and warning to prevent harm. We see a great potential to understand vehicles as single and cooperative MM systems, also for future autonomous cars. Whilst a large number of MM acquisition, processing, analysis and understanding techniques have been developed specifically for investigating comfort and safety applications, relatively few attention has been paid to the understanding of automobile-related MM content for security incident handling. MM systems can be a substantial enabler of immune systems for future automotive security to avoid/reduce harm by building *Multimedia-enabled Asimovian Secure Automobiles (MASA)*.

Acknowledgements: This work was partly supported by German Research Foundation, project ORCHideas (DFG GZ: 863/4-1). Thanks to A. Lang, S. Kuhlmann, S. Kiltz, M. Hildebrandt and J. Fruth for our joint work during the last six years.

REFERENCES

[1] T.Hoppe, S.Kiltz, J.Dittmann. 2009. Applying intrusion detection to automotive IT - early insights and remaining challenges. Journal of information assurance and security, Vol. 4. 2009, 3, ISSN: 1554-1010.

[2] T.Hoppe, S.Kiltz, J.Dittmann. 2011. Security threats to automotive CAN networks: practical examples and selected short-term countermeasures, In: Reliability engineering & system safety, Elsevier, Vol. 96.2011, 1.

[3] K.Koscher et al.., 2010. Experimental Security Analysis of a Modern Automobile. IEEE Symposium on Security and Privacy, 2010.

[4] D.Weld, O.Etzioni. 1994. The first law of robotics (a call to arms). Proceedings of the twelfth national conference on Artificial intelligence (vol.2), AAAI'94, ACM, USA, ISBN 0-262-61102-3.

[5] D.V.Pynadath, M.Tambe. 2002. Revisiting Asimov's First Law: A Response to the Call to Arms. ATAL '01, Springer, ISBN 3-540-43858-0.

[6] N.Schurr et al.. 2007. Asimovian multiagents: applying laws of robotics to teams of humans and agents. 4th international conference on Programming multi-agent systems, ISBN: 978-3-540-71955-7.

[7] S.Tuchscheerer, T.Hoppe, C.Krätzer, J.Dittmann. 2011. FirstAidAssistanceSystem (FAAS). Intelligent robots and computer vision XXVIII, Proceedings of SPIE; 7878.

Assure-It: A Runtime Synchronization Tool of Assurance Cases

Shunsuke Shida, Atsushi Uchida, Masaki Ishii, Masahiro Ide, and Kimio Kuramitsu
Yokohama National University
Yokohama, Japan
{shida-shunsuke-vn, uchida-atsushi-vt, ishii-masaki-cs}@ynu.ac.jp , imasahiro9@gmail.com, kimio@ynu.ac.jp

***Abstract*—More recently, the idea of runtime synchronization of GSN has been proposed, where evidences are being collected from logs that are produced by monitors and other software components. By introducing the runtime synchronization, GSN can be regarded as a program where its validity is checked by applying runtime contexts. In this fast abstract, we introduce Assure-It, a novel tool that enforces administration scripts with assurance cases guidance and runtime synchronization.**

Keywords—administration scripts; assurance cases; system dependability; software engineering supports;

I. INTRODUCTION

Scripting languages such as Bourne shell and Perl have been broadly used to perform system administration tasks, including system maintenance, system diagnosis, and failure response [1]. As these tasks are strongly related to the realization of system dependability (reliability, availability, etc.) requirements, software engineering supports for administration scripts are practically significant. However, most of these today's scripts are written in an ad hoc manner, unfortunately resulting in several causes of system failures.

Assure-It is a novel open source tool and developed in the JST/DEOS project to provide the means of modularizing scripting solutions for system administration under dependability requirements. The key idea is the use of assurance cases in order to associate dependability goals with administration tasks, which will be composed in a final executable script. Due to the specified association, the partial failure of script execution can be detected as an error from the associated dependability goals. In addition, Assure-It allows us to argue an incremental analysis of failure-case, which enables richer failure/error handling.

This fast abstract will show how Assure-It works with the concept of assurance cases. Several dependability goals (such as system and data availability, privacy and accountability) are argued over assurance cases, by associating monitoring and administration tasks. From the assurance cases, Assure-It can generate an executable script, directly connected to the realization of argued dependability goals.

The rest of this fast abstract proceeds as follows. Section 2 introduces Goal Structuring Notation, a standard notation of assurance cases, used in Assure-It. Section 3 overviews how Assure-It generates executable scripts from the arguments on assurance cases. Section 4 briefly describes the summary of this fast abstract.

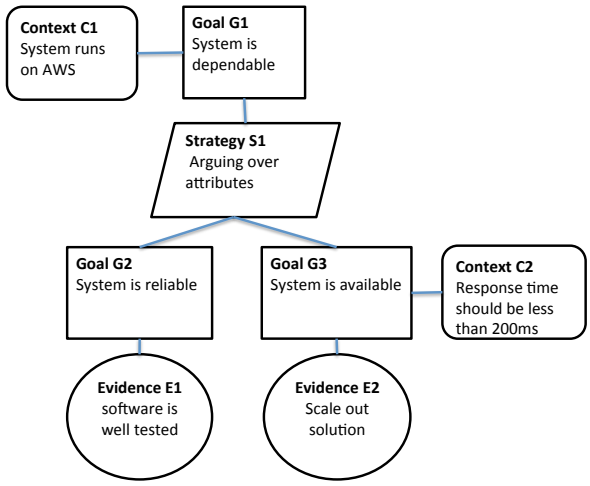


Fig. 1 An Example of GSN.

II. GSN AND THE CONCEPT OF ASSURE-IT

Assure-It has adopted GSN[2] as a common notation bridging existing assurance cases methodology. In addition, we attempt to add dynamic properties in order to synchronize assurance cases with runtime system through script executions.

A. Goal Structuring Notation

Goal Structuring Notation has four major elements, goal (depicted in rectangle), strategy (parallelogram), evidence (oval), and context (rounded rectangle). The goal element is a claim that a system certainly has some desirable properties. The evidence element is a fact supporting that the linked claim is true. The goal without linked evidence is called undeveloped goal and depicted with diamond. The strategy element is an assumption or a pre-condition that linked goal holds. Fig. 1 shows an example of GSN.

B. Extended Functional Evidence

The *functional GSN*, we propose in this fast abstract, is an extended one that accepts as one of valid evidence a program that produces logs supporting the correctness of its performance. Note that logging feedbacks are required for

runtime synchronization as depicted in Fig 2. A typical example of the program is a monitor notifying us of erroneous situations. Another typical example can be a system administration script that performs data backup and failure handling tasks, which straightforwardly lead to the realization of some dependability properties. From viewpoint of programming, these monitors and tasks are regarded as a function taking runtime contexts and then checking the validity of the associated goal.

It is important to note that the absence of failures in the functional element is not practical. Robin et al [3] proposes the meta assurance cases method to argue failure-case analysis on GSN. Using these methods, we can generate more reliable script that includes richer failure/error handling. Due to the space constrain, we are omitting the formalization of meta GSN in this fast abstract.

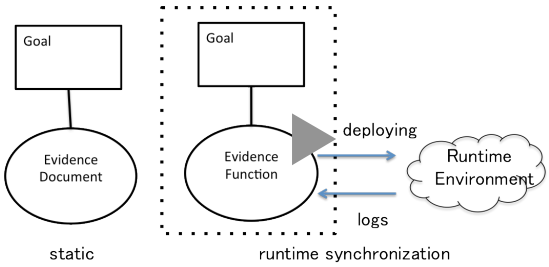


Fig. 2 Static Evidence and Runtime Synchronization

III. TOOL DESCRIPTION

Assure-It allows arguing the strategic division of dependability goals with assurance cases method. Fig. 3 describes the overview of Assure-It. It generates executable scripts from GSN arguments, binding each of operational solutions with strategy as control flows and deploys the generated scripts on running systems, and execution results are generated. Assure-It consists of two applications: a client and a server. Users create assurance cases by using client application, which is accessible through a mordern web browser.

A. Code Generation from Assurance Cases

As noted above, source code is generated from Assurance Cases, which is created by Assure-It to get in touch with runtime environment. In Assure-It evidence element consist of static evidence such as test results and code snippets, short length of shell scripts. The script that performs system administration is generated by making code snippets structured depending on rules described on strategy.

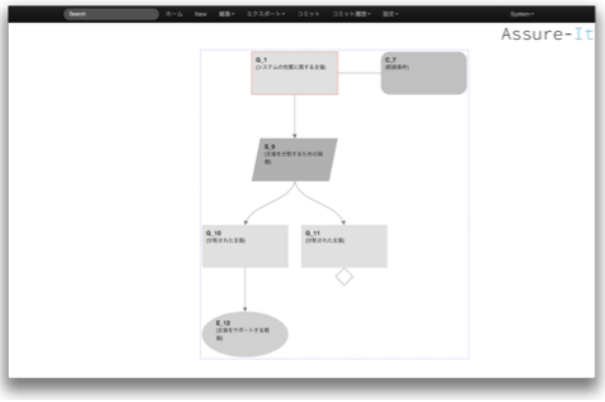


Fig. 3 An Overview of Assure-It

IV. SUMMARY

The assurance cases method provides the guidance to modularize administration tasks from viewpoint of dependability requirements. Assure-It is a tool that can generate an executable administration script by combining modularized tasks on assurance cases arguments. Running the generated script is an evaluation of dynamic assurance cases, being applied by runtime contexts.

REFERENCES

[1] Mario Tokoro. Open Systems Dependability: Dependability Engineering for Ever-Changing Systems. CRC Press, 2012.

[2] Tim, K. and Rob, W.: The Goal Structuring Notation A Safety Argument Notation, In Proc of DSN 2004 (2004).

[3] Robin E. Bloomfield, Peter Bishop: Safety and Assurance Cases: Past, Present and Possible Future - an Adelard Perspective, in Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, pp. 51-67, (2010).

Integrating agile practices into critical software development

Katarzyna Łukasiewicz, Janusz Górski
Gdańsk University of Technology
Gdańsk, Poland
{katarzyna.lukasiewicz, jango}@eti.pg.gda.pl

***Abstract*— Development of safety-critical software is constrained by the requirements of numerous standards and recommendations. In consequence, the development costs and time are considerably higher. In order to deliver high quality products faster and at lower cost safety-critical software developers may look for more efficient approaches and in particular the agile development practices are considered as a promising alternative. In this text we describe our research towards introducing agile practices into critical software development processes**

Keywords— safety-critical software; agile practices; software development; process improvement; safety assurance

I. PROBLEM STATEMENT

The need to deliver high quality systems, faster and at lower cost in comparison to competitors encouraged companies to look for more efficient solutions [1], [2]. Agile methodologies are known to successfully address these issues for non-critical projects. Presumably agile practices can reduce both cost and time to market when applied to safety-critical projects as well. While benefits can be significant, the main concern are quality and safety assurance. Plan-driven methodologies adequately address these objectives and have been integrated into the safety lifecycle for a long time. A growing body of evidence demonstrates that agile practices with their flexibility and the potential for shortening the development time and lowering the development costs could be complemented by more disciplined approach and therefore bring the best of the two worlds together [3], [4], [5], [6], [7], [8]. In particular, the challenge is to find an effective way of introducing agile practices into the critical software development process while ensuring that the level of assurance required by the corresponding domain specific standards is sufficient from the regulatory and system certification viewpoints. A mechanism for maintaining such control over the critical (agile) software development process could result in an explicit assurance (safety) case which integrates the assurance requirements with the arguments and evidence demonstrating that the requirements are satisfactorily implemented. Providing a methodological framework and tools for building, maintaining and assessing such assurance cases for the software development processes could help safety-critical software developers (in particular SMEs) to streamline their processes with agile practices and to maintain conformance with safety standards and certification requirements.

II. OBJECTIVES

The main goal of the research is to develop a comprehensive solution that would help safety-critical software developers to incorporate agile practices in the most profitable way while meeting the safety requirements imposed by standards and certification bodies. In particular, we are interested in a solution that provides for incremental development of an assurances case in paralel to the progress of the software development process which would make the assurances case a sort of 'side effect' of the software development.

Although we want the solution to be as generic as possible, we decided to focus on medical software domain. Medical devices are becoming increasingly software intensive and the market of suppliers as well as clients is growing rapidly. Health related electronic devices find their use in hospitals, homes, pharmaceutical companies and in many contexts has safety relevance. Most of such products need to be compliant with appropriate the related standards (e.g. GAMP [9], ISO 13485 [10], IEC 62304 [11]) and explicit assurance cases for medical devices are expected to be required by the relevant regulatory bodies.

III. METHODOLOGY

First, a literature review has been performed together with the review of the recommended, currently applied practices of critical software development and the review of currently used agile software development practices.

To better understand the risks related to application of the agile practices in critical software development processes we plan to perform a series of case studies with the involvement of software engineers. During these case studies the participants are requested to analyze and assess risks related to the agile practices and to propose the ways of controlling these risks. The case studies are bound by target system and its environment (the common insulin pump example has been selected for that purpose) and by the process context of software development (here, we concentrate on Scrum and Extreme Programming).

To represent the constraints imposed by the relevant standards and recommendations and to incrementally construct a related

assurance case we will use the TRUST-IT methodology [12] and in particular its application scenario related to standards conformance [13], [14] and the related platform of services [15]. Referring to standards and the present good practices we will develop a set of argumentation patterns that will also reflect the knowledge on risk mitigation related to the agile development practices acquired during the case studies. In order to justify each pattern we will prepare complementary meta-arguments. We described these ideas in more detail in [16], [17].

Upon completion of the planned case studies we will analyze the results and determine which of the agile practices raise most doubt when applied in safety critical projects and thus require extended evidence when building assurance arguments.

Alongside the case studies we will prepare templates for meta-arguments as well selected assurance argument patterns, for software development processes which incorporate agile practices, both presented in NOR-STA tool [15].

The complete method is planned to be a subject of validation with active participation of stakeholders and experts. Our aim is to establish cooperation with medical software developing companies to provide a satisfactory validation context.

IV. Expected results

The results of this research will be delivered as a set of argument patterns justified by associated meta-arguments supporting introduction of agile practices into critical software development and the related models of business processes explaining how the argument patterns are to be used to incrementally develop an assurance case for the resulting software. These results will be packaged on the top of the NOR-STA platform of generic services supporting application of evidence based argumentation [15].

V. Present state

The research described in this paper is an ongoing project. To date we carried out two case studies in 2012 (CS1) and 2013 (CS2) with the goal to investigate how junior software engineers identify and assess risks associated with applying selected agile practices to critical software development and what are their suggestions concerning risk mitigation. The results of the CS1 can be found in [16] and [17]. We plan to carry out workshops for more advanced practitioners in the nearest future as a continuation of the case studies.

References

[1] K. Petersen, C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," Empirical Software Engineering, vol. 15(6), pp. 654–693, 2010.

[2] M. McHugh, F. Mc Caffery, V. Casey, M. Pikkarainen, "Integrating Agile Practices with a Medical Device Software Development Lifecycle," Proceedings of European Systems and Software Process Improvement and Innovation Conference (EuroSPI), Vienna, Austria, 25-27 June, 2012

[3] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer., J. May. and T. Kähkönen, "Agile Software Development in Large Organizations," Computer, vol. 37(12), pp. 26-34, 2004

[4] H. Glazer, D. Anderson, M. Konrad and S. Shrum, "CMMI or Agile : Why Not Embrace Both!" Software Engineering Process Management – Technical Note for Software Engineering Institute, Carnegie Mellon University, 2008

[5] M. Poppendieck, T. Poppendieck, "Lean software development: an agile toolkit," Addison-Wesley, 2003

[6] J. Babuscio, "How the FBI Learned to Catch Bad Guys One Iteration at a Time," 2009 Agile Conference Proceedings,Chicago, USA, 24-28 August 2009, pp. 96-100

[7] N. Potter, M. Sakry, "Implementing Scrum (Agile) and CMMI together," Process Group Post Newsletter, 16(2), http://www.itmpi.org/assets/base/images/itmpi/Potter-ScrumCMMI.pdf, 2009

[8] M. Pikkarainen, A. Mantyniemi, "An Approach For Using CMMI in Agile Software De-velopment Assessments: Experiences From Three Case Studies," Proceedings of SPICE Conference, Luxembourg, 3-5 May 2006

[9] ISPE GAMP 5 Publications, http://www.ispe.org/index.php/ci_id/11614/la_id/1.htm

[10] ISO 13485:2003 , http://www.iso.org/iso/catalogue_detail?csnumber=36786

[11] IEC 62304, http://webstore.iec.ch/preview/info_iec62304%7Bed1.0%7Den_d.pdf

[12] J. Górski, "Trust-IT – a framework for trust cases", Workshop on Assurance Cases for Security - The Metrics Challenge. Proc. of DSN 2007, Edinburgh, UK, 2007, pp. 204-209

[13] J. Górski, L. Cyra, J. Górski, "SCF - a Framework Supporting Achieving and Assessing Conformity with Standards," Computer Standards & Interfaces, Elsevier, vol. 33, 2011, pp. 80-95

[14] J. Górski, A. Jarzębowicz, J. Miler, "Validation Of Services Supporting Healthcare Standards Conformance," Journal on Metrology and Measurement Systems, vol. XIX, No. 2, 2012, pp. 269-282

[15] NOR-STA project Portal, http://www.nor-sta.eu

[16] J. Górski, K. Łukasiewicz, "Agile development of critical software, can it be justified?" 7th International Conference on Evaluation of Novel Approaches to Software Engineering, Wrocław, Springer, 2012

[17] J. Górski, K. Łukasiewicz, "Assessment of risks introduced to safety critical software by agile practices - a software engineer's perspective," Computer Science, vol. 13(4), 2012

Secured Outsourced Linear Algebra

Amrit Kumar
Department of Computer Science
École polytechnique
Palaiseau, France
email: amrit.kumar@polytechnique.edu

Jean-Louis Roch
MOAIS, LIG-INRIA joint team
University of Grenoble
Grenoble, France
email: jean-louis.roch@imag.fr

Clément Pernet
MOAIS, LIG-INRIA joint team
University of Grenoble
Grenoble, France
email: clement.pernet@imag.fr

***Abstract*—We propose an interactive algorithmic scheme for outsourcing matrix computations on untrusted global computing infrastructures such as clouds or volunteer peer-to-peer platforms. In this scheme, the client outsources part of the computation with guaranties on both the inputs' secrecy and output's integrity. For the sake of efficiency, thanks to interaction, the number of operations performed by the client is almost linear in the input/output size, while the number of outsourced operations is of the order of matrix multiplication. The scheme is homomorphic, based on linear codes (especially evaluation/interpolation version of Reed-Solomon codes). Privacy is ensured by encoding the inputs using a secret generator matrix, while fault tolerance is ensured by the high error detection and correction capability of the code. The scheme can tolerate multiple malicious errors and hence provides an efficient solution beyond resilience against soft errors.**

I. Scenario

Computational power is often asymmetric. On the one hand, global computing platforms such as clouds available through internet provide a large scale of computational power and resources, but remain susceptible to various kinds of malicious attacks and faults. While on the other hand, a relatively weak local client is secure and reliable. The goal of today's computing infrastructure is to provide solution to draw the benefits of both of these components while ensuring secrecy of certain inputs.

In a scenario (Figure 1) where a weak but reliable client outsources a computation on a given input, the client should be able to efficiently verify the correctness of the result returned by the untrusted platform. Solutions for general computations either rely on Probabilistically Checkable Proofs (PCPs) [1], or fully-homomorphic encryption (FHE) schemes as in [5]. Considering their complexity, these constructions are currently beyond practical use. However, authors in [9] propose a new construction that can efficiently verify general computations under cryptographic assumptions. This construction compactly encodes computations as quadratic programs [4] which are then encoded as elements of a group equipped with a bilinear map. The weak client receives a proof (of constant size) along with the computational result. The verification procedure involves group operations.

Furthermore, on large scale computing systems error resilience is an issue. Error probability increases with the node count [2]. Algorithm-based Fault Tolerance (ABFT) [7] solutions have been explored for matrix computations without considering privacy. Focusing on a small rate of soft errors, an ABFT dense linear system solver is provided in [2] that is based on a low density parity check. Soft errors in general are produced in case the computing system is subject to cosmic radiations. Yet, on externalized computing platforms, malicious attacks that may corrupt a large number of intermediate computations are of concern. Such massive attacks occur due to Trojan attacks, and more generally orchestrated attacks against widespread vulnerabilities of a specific operating system that may result in the corruption of a large number of resources. In [10], an ABFT efficient solution for matrix multiplication is proposed for integrity against massive attacks that is based on a Reed-Solomon code.

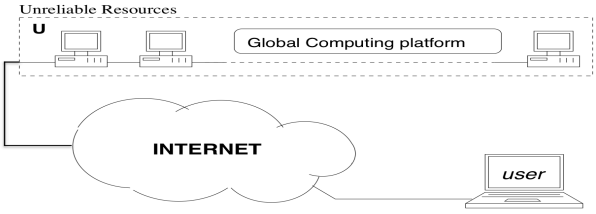


Fig. 1. Outsourcing a task to an untrusted platform

In this work we extend this solution in both directions. First, to provide secrecy, we use a secret code, based on a private Vandermonde generator matrix. Second, we extend the scheme to more general matrix computations such as LU factorization or matrix powering (with application to connected components in a graph), where the output data is non-linear in the input data. A major constraint is efficiency: trivial lower bounds for the cost of an interactive scheme are the size of the input and the output (memory cost) and the work of the best known algorithm (computational cost). We propose an asymmetric scheme for matrix multiplication that is almost optimal with respect to both the input/output size on the reliable resource (user side) and the best upper bound on the unreliable one (global computing platform side).

II. ABFT Dense Matrix Multiplication

As most of the linear algebra computations reduce to dense matrix multiplication, the design of the interactive zero knowledge protocol for computations is based on outsourcing matrix product. For the sake of clarity, we restrict in the sequel to $k \times k$ square matrices. The goal of these protocols is to keep the complexity of the operations almost linear in the size $O(k^2)$ of the input on the weak client, while on the unreliable cloud, a complexity $\widetilde{O}(n^\omega)$ would be acceptable,

This work has been partially supported by the LabEx PERSYVAL-Lab: n°. ANR-11-LABX-0025.

where $2 < \omega \leq 3$ denotes the exponent of matrix multiplication cost. A standard way for ABFT matrix multiplication consists in encoding the left and right operands by multiplying each by the generator matrix of a linear code [2], [7]. In the sequel, we use Evaluation-Interpolation linear codes, denoted RS (Reed-Solomon). These codes defined over a base field $\mathbb{F}$ are maximum distance separable. Assuming $\mathbb{F}$ larger enough, for any n with $\text{card}(\mathbb{F}) \geq n > k$, an (n, k) RS code is characterized by a $k \times n$ matrix G. A source vector x of size k is encoded by $y = x \cdot G$; any configuration of $(n-k)/2$ errors in y is guaranteed to be corrected.

The proposed secured ABFT protocol is as follows. The weak client initiates the protocol by generating two (n, k) RS codes, defined by G_1 and G_2. The input $k \times k$ matrices A and B are encoded as: $G_A = tr(G_1) \cdot A$ and $G_B = B \cdot G_2$, where tr denotes the transpose map. G_1 and G_2 are kept secret which eventually makes A and B secret. The client sends G_A and G_B to the global platform that performs the computation and sends back a result matrix R. Various errors may occur during computations or communications that are modeled by an insecure or noisy channel: the received matrix is seen as a perturbation of the correct encoded result $G_C = G_A \cdot G_B$. Upon decoding R, the client obtains a matrix $\widetilde{C}$ which verifies $\widetilde{C} = A \cdot B + E$, where $E_{n \times n}$ is the error matrix. The client can correct up to $(n-k)^2/4$ errors in $\widetilde{C}$ to recover $C = A \cdot B$.

The cost of computation on the reliable client sums to the cost of encoding and decoding. The encoding of each row or column reduces to k polynomial evaluations of degree k in n points, each computed in $O(n\log^2 n)$ with precomputation, so $\widetilde{O}(n^2)$ for the full matrices A and B. With fast extended GCD, decoding can be performed in $O(n\log^2 n)$ for each row or column, so $\widetilde{O}(n^2)$ for the matrix $\widetilde{C}$. The multiplication is performed by the remote platform in $O(n^\omega)$.

The client also has the possibility to verify the correctness of the result. This verification allows to detect cheating workers and even prevents man-in-the-middle (MITM) attacks. A cheating worker may not compute the matrix product correctly (and hopes that the fault remains undetected) while in the MITM scenario, an adversary might simply intercept the communication between the client and the platform and replaces the result by some other *good-looking* matrix, ex. the adversary might replace the matrix R by $G_A tr(G_A)$. For verification i.e. testing if $C = AB$, we propose to use probabilistic Freivalds' algorithm [3] which runs in $O(n^2)$, and states the correctness with good probability.

To ensure secrecy of inputs, the generator matrices G_1 and G_2 are kept secret. The evaluation points are randomly chosen and kept secret. However, if secrecy is discarded, the evaluation points are chosen to be $1, \alpha, \alpha^2, \ldots, \alpha^{\text{card}(\mathbb{F})-2}$, where α is a generator of the multiplicative group of the base field $\mathbb{F}$. With this choice of evaluation points, Fast Fourier Transformation (FFT) allows fast encoding and decoding and provides a logarithmic advantage. We note that a small field (such as $\mathbb{F}_2$) would not provide enough evaluation points.

III. Illustrative Example : Interactive Block LU Decomposition

Extending the idea, we also propose an ABFT interactive block LU decomposition protocol where the client outsources the task of LU decomposition to the platform. This protocol follows the standard block LU decomposition algorithm, where the inverse of the diagonal element is calculated locally and the blocks below the diagonal element are updated. The task of updating the sub-matrix to the right and below of the diagonal element is shared between the cloud and the client. We note that sub-matrix update requires the computation of matrix multiplication, hence is outsourced thanks to the previous matrix multiplication. Upon retrieving the product, the update operation reduces to addition and is performed locally. This interaction outsources the larger part of computation to the remote platform, while the smaller part is performed by the client. Let K be the block size: the cost of computation on client's side is $\widetilde{O}(n^2 K^{\omega-2})$ for block inversions and column update, and $\widetilde{O}(n^\omega K^{-\omega+2})$ for sub-matrix updates; while on the untrusted cloud, the cost is $O(n^\omega)$. For $\omega = 3$, the optimality is obtained when the block size is $K = \sqrt{n}$.

IV. Conclusion

In this paper, we design an efficient alternative to FHE based outsourcing with acceptable practicality and security. Our ABFT solution is resilient against malicious errors and hence goes beyond the correction of soft errors and can even handle MITM attacks. The scheme computes matrix operations such as matrix-matrix multiplication and can be extended to interactive protocols performing more complex operations on matrices. The ongoing work includes quantifying security provided by our scheme, the study of the related cost-security trade-off and include other computations in the framework. While this work is based on large finite fields, a perspective is the design of efficient solution for floating point numbers, based on dedicated encoding for matrix multiplication.

References

[1] Sanjeev Arora. Probabilistic checking of proofs: a new characterization of np. In *Journal of the ACM*, pages 2–13, 1998.

[2] Peng Du, Piotr Luszczek, and Jack Dongarra. High performance dense linear system solver with resilience to multiple soft errors. In *ICCS*, pages 216–225, 2012.

[3] Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP Congress*, pages 839–842, 1977.

[4] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps.

[5] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[6] Philippe Golle and Ilya Mironov. Uncheatable distributed computations. In *Lecture Notes in Computer Science*, pages 425–441. Springer, 2001.

[7] Kuang-Hua Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33(6):518–528, June 1984.

[8] Fabian Monrose, Peter Wycko, and Aviel D. Rubin. Distributed execution with remote audit. In *In Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, pages 103–113, 1999.

[9] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. Cryptology ePrint Archive, Report 2013/279, 2013. http://eprint.iacr.org/.

[10] Jean-Louis Roch and Sebastien Varrette. Probabilistic certification of divide & conquer algorithms on global computing platforms. application to fault-tolerant exact matrix-vector product. In ACM publishing, editor, *Parallel Symbolic Computation'07 (PASCO'07)*, London, Ontario, Canada, July 2007.

[11] Radu Sion. Query execution assurance for outsourced databases, 2005.

Security-Informed Safety

"If it's not secure, it's not safe"

Robin Bloomfield
Centre for Software Reliability
City University London
reb@csr.city.ac.uk

Robert Stroud
Adelard LLP
London, UK
rjs@adelard.com

***Abstract*— Traditionally, safety and security have been treated as separate disciplines, but this position is increasingly becoming untenable and stakeholders are beginning to argue that if it's not secure, it's not safe. The idea of combining safety and security is not new but neither is it straightforward. To illustrate the complexities of the problem, we explore some of the challenges that need to be overcome in order to develop a principled approach to "security-informed safety".**

Keywords—security-informed safety; assurance cases.

I. INTRODUCTION

For a system to be safe, it also has to be secure. Otherwise, a safety critical system – one that can harm or injure people – could provide attackers with a potential mechanism for causing widespread damage or panic, and it is credible that such systems could become the target of malicious actions.

In principle, achieving interworking between safety and security should be straightforward. Both are sophisticated engineering cultures that emphasise the need for good process, the importance of risk analysis and the need for assurance and justification. However, these similarities are superficial and in practice there are significant challenges, as experience with large-scale systems has shown.

To illustrate the complexities of the problem and in order to stimulate debate, we explore some of the technical issues involved in combining safety and security assurance in a principled way. Our work is informed by an investigation of these issues in the context of the railway industry [1], but is more widely applicable to safety engineering in general.

II. CONCEPTS

The commonalities between safety and security are frequently obscured by the use of different concepts and terminologies. Indeed, there is considerable variation in terminology both within and between the safety and security communities. Thus, to achieve a shared understanding of the key concepts within each domain, there is a need to establish a lingua franca or even a common ontology.

The IFIP WG 10.4 dependability taxonomy [2] offers some hope for defining a consistent set of terms. In particular, it makes a clear distinction between cause and effect and highlights the need to be clear about system boundaries.

Broadly speaking, safety is concerned with protecting the environment from the system whereas security is concerned with protecting the system from the environment. Security and safety can both be viewed as kinds of dependability and use similar techniques to identify potential failure modes and assess their impact on the overall system. Thus, there is considerable overlap between safety and security methods, although the focus is different and in some cases safety and security requirements can be in conflict.

In particular, one of the major differences between safety and security is that a secure system needs to cope with evolving threats and changes to the environment through design and architectural measures as well as operational ones. It is important for the system to remain safe and secure despite such changes, in other words, to be resilient to change.

III. PRINCIPLES

There are many overlaps between safety and security principles, but there are also some significant differences in emphasis and some potential conflicts. For example, defence in depth is an important architectural principle for both safety and security that depends on the use of multiple, and as far as possible independent, barriers. However, security considerations are likely to challenge the effectiveness and independence of safety barriers.

From a safety system perspective, security principles [3] such as economy of mechanism, least privilege, and psychological acceptability are probably all readily acceptable. Other principles, such as complete mediation and end-to-end arguments, could have a significant impact on the architecture and performance of systems. But perhaps the most radical security principles from a safety perspective are those based on Kerchoffs' principle [4], namely ease of recovery and open design.

In particular, although safety systems are already designed to support operational changes for calibration and maintenance, the ease of recovery principle, which states that the security of the system should not depend on anything that cannot be easily changed, could have far reaching impact on the architecture of safety systems.

Moreover, changes to threats over the lifetime of the system will probably mean that controls that were adequate initially will need to be reconsidered. This has implications for

This research was commissioned on behalf of UK government and the UK rail industry.

the architecture and lifecycle of embedded safety systems where design life may be 20-40 years.

Given the uncertainties around future threats, systems should be designed to be adapted and replaced perhaps sooner then would be necessary from just a safety perspective. This could have significant architectural and cost implications for large infrastructure projects, particularly those that are already in progress

IV. METHODOLOGY

Risk assessment is a fundamental step in safety and security analysis, but the underlying threat model is different. There is a need for a unified methodology for assessing the threats to the safety and security of a system.

Security considerations can have a significant impact on a safety case. For example, there needs to be an impact analysis of the response to security threats and discovery of new vulnerabilities and reduction in the strength of protection mechanism. This suggests a greater emphasis on resilience of the design.

It is also necessary to consider the potential for attack during a safety incident and the opportunity this might provide for malicious activity. A fail-safe state may not be as safe as previously thought if the system is under attack and the assumption that any security attack on a control system could only, at worst, cause a fail-safe state to be reached is in general not true. Moreover, assumptions about the capabilities and state of society may change; for example, consider managing a safety incident during a major security incident.

Given the importance in security of open scrutiny of design and implementation (e.g. of crypto), it is an appropriate question whether security-informed safety cases in entirety or part should be disclosed. Within the safety community, the principle of independent assessment is well established, but the design details within a safety case are usually considered to be confidential and are not made public in their entirety.

The key question is whether publishing the detailed design and safety analysis for a system would make the system less or more safe and secure, or more precisely which aspects would it be beneficial to expose and which not.

V. STANDARDS

Safety standards already require *"malevolent and unauthorized actions to be considered during hazard and risk analysis"* [5], and there have been a number of domain-specific attempts to define a unified approach to safety and security assurance [6][7]. However, the standards framework for dealing with security-informed safety needs to be more explicitly designed than is currently the case. In particular, the relationship between generic and domain-specific safety and security standards needs to be clarified, and terminological and conceptual differences need to be resolved.

The standards framework should be based on explicit principles and use a consistent terminology. The standards groups should ensure they have available a suitable mix of both security and safety expertise.

Standards often use "levels" as a way of classifying systems, risks and controls. However, it is important to understand the assumptions that underpin these classification schemes and not to confuse different kinds of classification. In particular, risk levels, requirement levels, and assurance levels need to be carefully distinguished.

A particular concern is the problem of justifying requirements that specify the use of particular methods and tools to achieve a specific level. In order to support interworking between safety and security standards, we need to develop a better understanding of the rationale for such recommendations and the evidence base that supports them.

Security standards are often based on security controls, a concept that embraces a wide range of different interventions covering process, product and organisation. In contrast, safety standards are typically based on an engineering life cycle model. In principle it should be possible to relate safety mitigations to security controls, but in order to perform such an analysis, it will be necessary to define a common way of classifying controls and mitigations.

VI. NEXT STEPS

We believe that some or all of the following next steps would be helpful in establishing a more principled approach to "security-informed safety".

- Develop guidance on concepts and terminology to support dialogue between the safety and security communities.
- Research the applicability of security principles to safety and the associated trade-offs and conflicts.
- Consider how to make credible arguments that safety and security risks have been reduced to as low as reasonably practicable.
- Investigate how a Claims-Arguments-Evidence based methodology could be used to support the development of a security-safety protection profile.
- Intercept and support the standards process in order to clarify the relationship between safety and security and resolve some of the terminological and conceptual issues.

REFERENCES

[1] R. E. Bloomfield and R. J. Stroud, "Safety and Security: Concepts, Standards and Assurance", Adelard reference D/719/138002/2, v2.0, March 2013.

[2] A. Aviziensis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, January-March 2004.

[3] J. H. Saltzer, M. D. Schroeder, "The Protection of Information in Computer Systems" CACM Vol. 17(7), July 1974.

[4] A. Kerckhoffs, 'La cryptographie militaire', *Journal des sciences militaires*, vol. IX, pp. 5–38, Jan. 1883, pp. 161–191, Feb. 1883.

[5] EN 61508-1:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 1: General Requirements.

[6] Praxis High Integrity Systems, SafSec: Integration of Safety & Security Certification, November 2006.

[7] ED-202, Airworthiness Security Process Specification, EuroCAE, December 2010.

Adequacy of contract grammars for component certification

Alejandra Ruiz, Huascar Espinoza
ICT-European Software Institute Division
TECNALIA
Zamudio, Spain
{alejandra.ruiz, huascar.espinoza}@tecnalia.com

Tim Kelly
Department of Computer Science
University of York
York, United Kingdom
tim.kelly@cs.york.ac.uk

***Abstract*— The use of contracts in component-based development is a well-established approach. However there exists a wide range of views as to the nature of the contracts that are necessary to support safety-critical systems development, assurance and certification. Different standards and projects have tried to reduce ambiguity and propose the best practice in this area. In this paper we present work that moves one step further forward with the creation of a methodology and grammar that incorporates encompasses and helps structure current models of 'safety contracts'.**

Keywords— component; certification, contracts, grammar

I. INTRODUCTION

As systems become increasingly complex and distributed development becomes increasingly commonplace, there has been greater interest in component-based and contract-based approaches to system development and assurance. In parallel with this, certification standards are increasingly supporting the the notion of modular (component) certification. For example, DO-297 [1] addresses this topic in the context of modular avionics and the concept of SEooC (safety element out of context) has been introduced in the new automotive safety standard ISO 26262 [2]. These new concepts are not easy to apply. Ruiz et al [10] has previously described the difficulties faced by industry when attempting to apply the SEooC concept (particularly with respect to managing assumptions). Contract based approaches can help structure and manage the activities associated with compositional certification. .

II. TECHNICAL APPROACHES

A number of different technical approaches to contract specification have been studied. Each of them typically focuses on solving one objective. There are some identified reasons behind formalizing contracts such as [11]:

- Avoid human errors
- Support for validation or checking
- Interoperability between different suppliers
- Facilitate the integration of the components within the system

The following table shows some of the approaches already explored for improving the definition of contracts:

TABLE I. DIFFERENT CONTRACT TECHNICAL APPROACHES

Approach	**Description**	***Ref***
Formal language	Specification of a formal meta-modeling language for design contracts. It provides information about components behaviour, variables and interfaces but not the implementation	[3]
	Specification of a formalization of safety cases. Safety argumentation can be logical deduction, probabilistic, expert judgement or historical experience. Formalizing some elements supports precision and cheching methods	[12]
Metamodel	The 'Rich Component' Metamodel focuses on the integration of component-based design by the use of contracts from different perspectives: such as operational actors, functions, logical components or technical components.	[6]
Reference architecture	In different domains there have been initiatives to define a reference architecture with an open API e.g. AUTOSAR. These reference architectures can be decomposed into different components. The integration of these components is implementation independent and is aided by well-defined interfaces	[7] [13]
Properties modelling	Formal and structured property modelling .	[8]
Pattern	Definition of a generic pattern for safety case contracts. They propose the GSN notation as a way to structure agreements between safety case modules.	[9]

All of these approaches try to solve parts the whole problem from different perspectives. Some approaches, such as those that concentrate on defining reference architectures, focus on design standardization and component integration rather than certification. (Although an argument can be made that they may reduce the costs of certification through establishing standardized interfaces.)

III. HIGH LEVEL GUIDANCE

Different assurance and certification standards have addressed the problem of component-based assurance in different ways. Here, we focus especially on the avionics and

automotive domains. In the automotive domain, the introduction of the Safety Element of Context (SEooC) together with the standard ISO 26262 [2] has opened the door to modular approaches regarding functional safety. An example of a safety-oriented 'contract' can be seen in ISO 26262 [2], where the term Development Interface Agreement (DIA) is used to defines the procedures and responsibilities allocated within distributed developments for items and elements. In the DIA the supplier should exchange with the customer information such as: feedback about conflicts, completeness, consistency, etc.; technological limitations, behaviour models, incl. fault models, feedback about boundary between the component and its environment.

In the avionics domain we can find similar requirements with respect to module and application reuse within an IMA (Integrated Modular Avionics) platform. In DO-297 [1] (amongst other requirements) it is required that limitations, assumptions, etc. are documented and a usage domain analysis performance to ensure that any component is being reused in the a way that is compatible with the original design intent.

Other aerospace avionics guidelines such as AC 20-148 [4] concerning reusable software components indicate that in order to reuse components, stakeholders must identify any installation, safety, operational, functional and performance possible concerns. Developers need to state clearly the DO-178B objectives that are fully and partially addressed, and how compliance has been achieved. They need to state clearly the failure conditions, safety features, protection mechanism, architecture limitations, software levels, interface specification and the process for certification. AC 20-170 [5] defines incremental acceptance as, "A process for obtaining credit toward approval and certification by accepting or finding that an IMA module, and/or off-aircraft IMA system complies with specific requirements. This incremental acceptance is divided into tasks. Credit granted for individual tasks contributes to the overall certification goal." This definition implies that the process in which the system assurance is performed is also important. At every stage some form of recognition is submitted in relation which a compliance data package. The process is divided into 6 tasks: Module acceptance; Application acceptance; IMA system acceptance, Aircraft integration of IMA system, Change and reuse of modules or applications. Reuse can be done at Task 1and 2 level.

IV. COMPARISONS

Our on-going work addresses the challenge of integrating the existing approaches described in the previous sections. In doing this, we hope to improve consistency of approach across and reduce uncertainty as to the necessary considerations in safety-oriented contract specification and management.

Guidelines from the standards offer the best practices and interpretations of the standards in order to comply with certain requirements. Those best practices can be modelled within the different technical approaches and impact on the methodology for the system development. Different technical measures can be put into place in order to assure the correct and complete following of the guidance and practices.

In our approach we propose to formalized contracts through an well defined and structured contract 'grammar' to support how users may systematically assure safety of their system while integrating components. In order to do it we propose the definition of a BNF (Backus Normal Form or Backus–Naur Form) grammar. In this structure we will take into account the different views of contracts. AC 20-148 states that, "identify any installation, safety, operational, functional, or performance concern". We organise our contract grammar around these aspects to help identify such concerns. Fenn [9] proposes to use argumentation not only on safety cases but also on safety contracts, so our grammar should support argumentation. Rusby [13] has previous identified different types of argumentation. These types can be used to help provide extra structure to the argumentation aspects of the contract grammar.

One of the benefits of formalizing safety contracts will be the possibly of tool support for checking or generating contracts. We are using Xtext [14] as the technology to implement our grammar and be able to interoperate with other future tools. Moreover, with the provision of a defined grammar for safety contracts we will be able to support validation of contracts (e.g. helping identify incomplete contracts).

ACKNOWLEDGMENT

The research leading to these results has received funding from the FP7 programme under grant agreement n° 289011 (OPENCOSS) and nº608945 (Safe Adapt)

REFERENCES

[1] RTCA DO-297/EUROCAE ED-124 Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations

[2] International Organization for Standardization (ISO), ISO26262 Road vehicles – Functional safety, ISO, Nov 2011

[3] D.2.5.4 Contract Specification Language (CSL); SPEEDS Project; Deliverable; Rev. 1.0.1; April 2008: URL: http://speeds.eu.com/downloads/D_2_5_4_RE_Contract_Specification_Language.pdf; PDF-Document; Last visit: 2013-02-13

[4] FAA Advisory Circular: AC 20 148 Reusable Software Components

[5] FAA Advisory Circulation AC 20-170

[6] D_SP1_R3.3_a_M3 Meta-Model Concepts for RTP V; CESAR Project; Deliverable. http://www.cesarproject.eu/index.php?id=47&L=0; PDF-Document; Last visit; 2013-02-12

[7] Fürst S.: AUTOSAR – An open standardized software architecture for the automotive industry. 1st AUTOSAR open conference & 8th AUTOSAR premium member conference, Detroit, US, Oct. 2005

[8] ATTEST2 Project, URL: http://www.atesst.org Last visit: 25/06/2013

[9] J. Fenn, R. Hawkins, P. Williams, and T. Kelly, "Safety Case Composition Using Contracts -Refinements based on Feedback from an Industrial Case Study," in Proceedings of 15th Safety Critical Systems Symposium(SSS'07), February 2007

[10] A. Ruiz, H. Espinoza, F. Tagliablò, S. Torchiaro, A. Melzi, "A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance" Proceedings of 21st Safety Critical Systems Symposium, February 2013

[11] Machine-checkable Assurance Case Language http://www.omg.org/cgi-bin/doc?sysa/2012-9-4

[12] J. Rushby, "Formalism in safety cases," in Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, Springer, 2010, pp. 3–17.

[13] ARINC 653 Avionics Application Software Standard Interface

[14] Xtext, » http://www.eclipse.org/Xtext

Communication integrity for slow-dynamic critical embedded systems

Amira Zammali [(1),(2)]
[(1)] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[(2)] Univ of Toulouse, UPS, LAAS, F-31400, Toulouse, France
Email: zammali@laas.fr

Agnan de BONNEVAL [(1),(2)]
[(1)] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[(2)] Univ of Toulouse, UPS, LAAS, F-31400, Toulouse, France
Email: agnan@laas.fr

Yves CROUZET [(1),(2)]
[(1)] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[(2)] Univ of Toulouse, LAAS, F-31400, Toulouse, France
Email: crouzet@laas.fr

***Abstract*—We present, in this paper, challenges and works in progress for a new communication integrity approach that is based on error detection codes and targets slow-dynamic critical embedded systems. The novelty of this approach lies in the fact that it takes profit of the fault tolerance criterion of slow-dynamic systems. Thus, it does not focus on each exchanged message but rather on a set of messages (which number is being be set according to the safety requirement of the targeted system). This approach relies on a set of control functions whose error detection capabilities and coverage are complementary, which improves the resulting detection capability compared to the usual use of one unique control function.**

Keywords—slow-dynamic systems, critical embedded systems, fault tolerance, safety, communication integrity, error detection codes.

I. Introduction And Problematic

Nowadays, critical embedded systems are based on complex networks including active intermediate nodes. This increases the occurrence of erroneous messages and introduces new types of errors, even though the occurrence of undetected erroneous messages can lead to catastrophic events (e.g. airplane crash). Thus, ensuring the communication integrity in such systems is crucial. Traditionally, integrity policies aim at avoiding the occurrence of one undetected erroneous message. So they use heavy error detection codes in order to obtain an efficient detection power per each exchanged message. Yet, previous works [1] in our research team revealed that, for some kinds of systems, to meet the safety requirement, there is no need to focus very strongly on the integrity of each message. In fact, avoiding the occurrence of a number X (X>1) of undetected erroneous messages among N messages is sufficient for these systems. These previous works have defined a cumulative error detection policy consisting of a set of complementary control functions. This policy was based solely on CRCs codes and targeted Flight Control Systems. These works open horizons to us in order to dig deeper and propose a more complete and generic approach adopting the complementary property of used functions. Section II describes the targeted systems: slow-dynamic critical embedded systems. Section III presents the context communication integrity approach to be adopted in these targeted systems and section IV is devoted to present our works in progress.

II. Slow Dynamic Critical Embedded Systems

The class of systems we target in our works is the class of slow-dynamic critical embedded systems. The critical property induces high safety requirements. It means the system is low fault tolerant because of some kinds of failures may lead to catastrophic events (loss of goods and even lives): typically, failure rate must be less than 10^{-9} failure/hour. "Embedded" means that such systems do not dispose of a huge of resources (memories, processors, etc.) and communications are based on short messages (e.g. 100 bits for Flight Control Systems).

The novelty, here, is the "slow-dynamic" property of the system (first defined in [1]). In fact systems can be classified into two classes: i) fast-dynamic systems; ii) slow-dynamic systems. "Fast-dynamic systems" are defined by a duration of their significant changes very close to the duration of the refresh cycle of their changes command computation. This enables to send one unique message (command) during the duration of significant change. Thus, an undetected erroneous message may lead to a catastrophic event. While the so-called "slow-dynamic systems" are defined by a duration of significant changes is much larger than the refresh cycle duration. This enables to send several messages (commands) during this duration (see Fig.1). Thus, a catastrophic event cannot result from one undetected erroneous message, but only from a set of undetected erroneous messages whose number exceeds a threshold being set according to the case study.

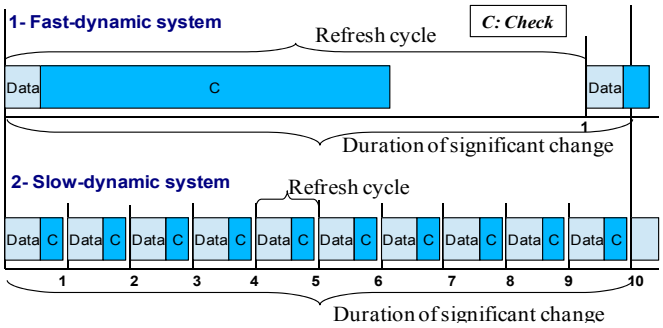


Fig. 1. Slow-dynamic systems compared to fast-dynamic systems

An example of slow-dynamic system is the flight control system on commercial airplanes. Computers exchange control-command messages with the actuators governing the flight

control surfaces. These surfaces are designed to move slowly (about 50°/s). The refresh cycle duration (about 10 ms) is much smaller than the duration of a significant change. Thus, many control-command messages are exchanged and the system can tolerate several undetected erroneous messages.

This slow-dynamic property makes possible to deal with the problem of communication integrity in a different way.

III. Communication Integrity Approach

As described before, for slow-dynamic critical embedded systems, the integrity policy does not focus on each exchanged message. Our goal is rather to avoid the occurrence of more than X undetected erroneous messages among N transmitted ones. So instead of using one unique control function, we rely on a set of complementary ones, which means they have complementary (therefore cumulative) detection capabilities and coverage. So, this policy is more efficient, as described in Fig. 2, where we assume: all messages are erroneous, X=3, N=10, three complementary functions F1, F2 and F3 (and D: Detected, ND: NonDetected).

In our approach, we target the application layer and we aim at ensuring the end- to-end integrity. We consider the following assumptions:1) the key safety requirement of considered systems is the tolerance of less than 10^{-9} failures per hour [2] [3]; 2) the X undetected erroneous messages among N can be considered either as consecutive, nor as not consecutive; 3) intermediate nodes are active (with memories and treatment capabilities); 4) communication channels are binary and symmetric; 5) messages size is around 100 bits; 6) refresh cycle is around some ms; 7) targeted errors are random independent errors, burst errors and particularly repetitive errors; 8) the redundancy must be as low as possible in terms of check bits, networks components and channels.

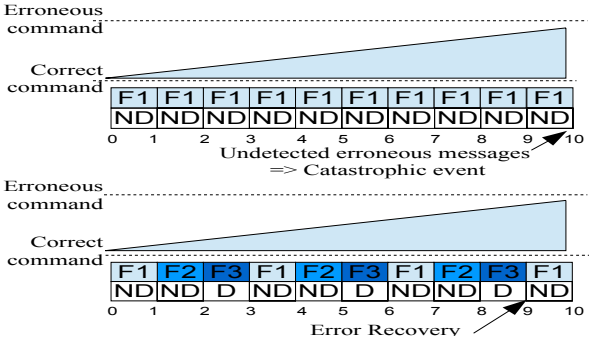


Fig. 2. Error detection policy : one function versus multi functions policy

IV. Challenges And Works In Progress

Several challenges are arising with the specificities of our error detection policy. The first one is to find theoretical proofs and/or validate by experimentations the complementarity property of nonCRC codes. Previous works [1] have proved that two different CRC generator polynomials are complementary only if they share a minimum of common factors. Now, our goal is to find other complementary codes. An other side of this challenge is to use complementary codes belonging to different families in the same detection policy. The second challenge is to limit the resources consumption in terms of calculation time and memory since we consider embedded systems. The third challenge is to decrease the redundancy in terms of check bits in order to respect the short messages property we consider. The forth challenge is to ensure the synchronization between the network nodes (source, sink and intermediate nodes) in order to be sure that they use all, at every refresh cycle, the same control function. The last challenge is to extend the application domain of the detection policy and explore other slow-dynamic critical embedded systems and not only be limited to flight control systems.

To take up these challenges, our approach is based on an optimal error detection codes selection. In fact, for CRC codes, it was proven [4] that conventional polynomials are not necessarily the best choice to make. Moreover, we are exploring lightweight codes like Adler and Fast CRCs that have efficient capabilities with a lower complexity than CRCs. Besides, we are working on automotive systems in order to study the possibility to apply our policy to it. To validate our theoretical solution, we will rely on simulations via the Matlab-Simulink platform which provides tools to model and simulate communications. Experimentations will consist of Monte Carlo simulations. We have started modelling our experimentations. We are working on three models of errors injection: i) exhaustive injection (considering all possible erroneous messages); ii) selective injection (considering a kind or a subset of erroneous messages) and iii) random injection. To accelerate simulations, we are exploring the "Parallel computing" tool, a Matlab tool that we are working on it. It permits to make a set of parallel simulations while providing the synchronization between inputs, outputs and parameters.

V. Conclusion

In this paper, we have first presented how the dynamic of a system can impact the way of considering communication integrity in critical embedded systems. For the class of slow-dynamic systems, we have reminded an innovative solution based on complementary error detection functions. In this context, one of the most important challenge we deal with, is to find (by theory or simulation) other error detection codes (than CRCs codes previously used), that would consume less time and memory, while having the property of complementary detection capabilities and coverage. And we seek to extend this approach to other domains than only aeronautic.

References

[1] A. Youssef, Y. Crouzet, A. de Bonneval, J. Arlat, J. J. Aubert and P. Brot, "Communication integrity in networks for critical control systems". The European Dependable Computing Conference (EDCC), Coimbra, Portugal, 18-20 Oct. 2006, pp.23-34

[2] M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico and Ph. Koopman, "Coverage and the use of cyclic redundancy codes in ultra-dependable systems". The international Conference on Dependable Systems and Networks (DSN), Yokohama, Japan, 28 June-1 July 2005, pp.346-355

[3] Federal Aviation Administration, "System Safety Handbook, chapter 3:Principles of System Safety", 30 december 2000, 19 p.

[4] Ph. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks". The international Conference on Dependable Systems and Networks (DSN), Florence, Italy, 28 June-1 July 2004, pp.145-154.

Robust by „Let it Crash“

Christoph Woskowski, Mikolaj Trzeciecki, Florian Schwedes
Zühlke Engineering GmbH
Landshuter Allee 12
80637 Munich, Germany
{christoph.woskowski,mikolaj.trzeciecki,
florian.schwedes}@zuehlke.com

Keywords— safety-related; fault-tolerance; supervisor hierarchies; let-it-crash; Erlang

A. Introduction

Critical software systems are bound to perform extensive error detection and exception handling. The corresponding source code is typically implemented in a defensive programming style. Typical strategies to ensure robustness include elaborate exception handling and error-code returning routines. Most often, error handling code fragments are often not separable from the source code realizing the core functionality, and they are prone to errors themselves. For extending exception handling in order to further improve fault-tolerance, even more source code is necessary. However some leftover vulnerability always remains, especially in complex, multithtreading, and distributed systems. Producing more code ultimately results in more complexity while reducing readability and maintainability. This in turn inevitably leads to programming errors.

The programming language Erlang breaks a new ground for handling fault-tolerance problems. Very light-weight processes in separate memory areas enable straightforward concurrency with communication solely based on message passing. Processes are able to monitor and – in case of a process termination – restart each other very swiftly. The exception handling method of choice for a worker process is to terminate itself (“let it crash” – LiC), if it is unable to handle the situation locally. Dedicated supervisor hierarchies ensure appropriate error responses by starting a different process or by restarting a new instance of the terminated one.

This work presented in this abstract investigates, whether the let-it-crash paradigm for fault-tolerant systems may also be applicable to safety-related software projects. The scenario chosen for this demonstration approximates (and simplifies) a project within the medical device control software domain.

B. ModelProject

Although often a necessity, long term hospitalization is expensive and can even pose a health threat to hospitalized people. For reducing these costs and risks, a number of patients are treated at home. In such a case, an appropriate and reliable monitoring system must be used. In our (fictional) project, such a monitoring system is developed which uses so called “functional clothing”. This clothing is a kind of garment incorporating wireless sensors, which allows the patient to move freely around without being restricted, even while their vital signs keep being monitored. The signals from the sensors arrive wirelessly at a base station located in the same house or room as the patient. This device employs a constant connection with all active sensors, is able to power them on and off and switches to an alternative measurement location if need arises (failure, implausible data). The base station establishes a connection with the hospital and transfers the data for evaluation.

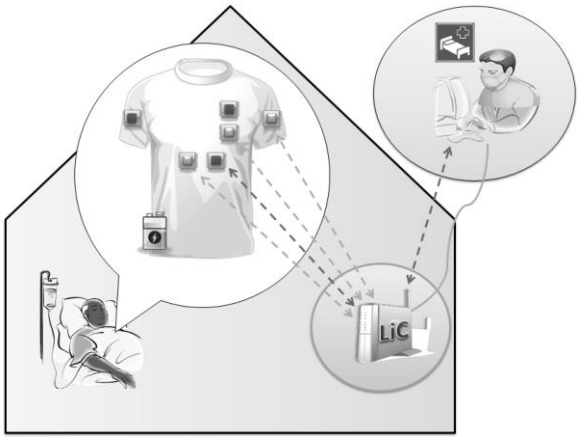


Fig. 1. Proof of concept scenario

The subject matter of the LiC proof-of-concept is the software development for the base station. The project focusses a high reliability of measurement data acquisition and transfer of the patient’s vital signs to the hospital. A maximum number of currently active sensors is set to limit power usage. At the same time a minimum coverage of the vital signs has to be guaranteed: for every point in time at least two out of three critical values (heart rate, breathing rate and blood pressure) have to be available.

The safe state of the house station is a complete shutdown, since the hospital system gets alarmed about the missing data.

C. Implementation and testing

Our prototypical implementation in Erlang makes use of the supervisor hierarchies and allows for deployment of worker processes and supervisors as well as the evaluation of separating business logic from error handling. The

development concentrates on the software of the base station and just simulates the external sensors on the one side and the hospital system on the other. The diagram [Fig. 2] depicts the example setup, showing the runtime view of the processes and dependencies.

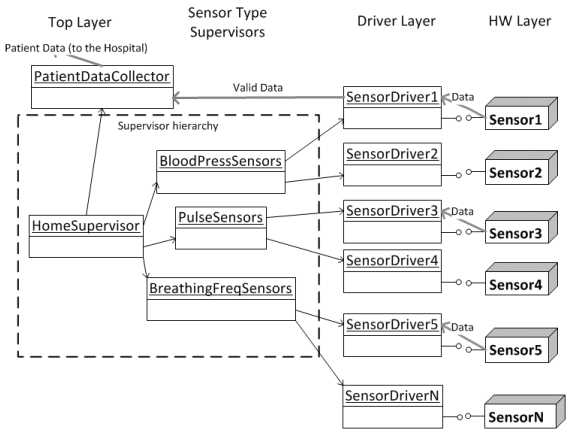


Fig. 2. Runtime view of processes and dependencies

The generic supervisor hierarchy is solely responsible for creating the worker processes (sensor drivers and data collector) and for handling errors by restarting or replacing terminated processes.

The sensor drivers and the data collector on the other hand contain the core functionality (business logic) and no error handling at all. In case of missing sensor values, for example, the sensor driver just terminates and gets replaced. The same happens if there is data available but outside of valid limits.

In connection with regulatory requirements concerning medical devices (e.g., IEC 60812), we test the prototype depicted above for the following failure situations:

- Failure to perform the desired function
- Performing a function that was not desired
- Performing a function at a wrong time
- Incorrect timing or order of executions
- Recognition and handling of critical conditions by the system

A simple and effective variant of testing fault-tolerance is based upon a so called "Chaos Monkey" - a process injected into the system under test with the sole task of randomly terminating other system processes. In traditional systems with a small number of complex tasks, this typically leads to complete failure within a very short period of time.

In our system following the LiC philosophy this only triggers the process monitoring and thus a fast replacement of the terminated software part. This has been tested in a simulated uninterrupted Base Station run of multiple days. In spite of the chaos monkey killing random components, our system is able to maintain basic functionality.

Further, we tested the concurrency behavior of the system by adding the necessity of the sensors to calibrate themselves. The calibration functionality opposes the normal sensor activity, as the abovementioned limitations to the maximum and minimum count of the active sensors remain in place. In our prototype, a sensor performing calibration at undesired moment gets "crashed" by a dedicated supervisor, following the LiC approach consequently.

D. Conclusion

Considering the LiC application hypotheses proposed above, the following can be stated about the patient monitoring scenario implemented in Erlang:

1. ***Ensure the execution of critical functionalities***. Ill-performing tasks are stopped and restarted, no matter the cause. For instance, a malfunctioning sensor driver gets terminated and replaced by another one.

2. ***Prevent the unintended execution of a function***. When a functional monitor detects a worker executing an unintended function, this worker gets terminated and replaced, thereby preventing the execution. For instance, a sensor calibration is aborted when there is another calibration request of higher priority.

3. ***Define and monitor the conditions for carrying out a critical function***. Workers and functional monitors can control task execution and results given distinct validation checks. This excludes any measures to correct the situation besides restarting affected processes. A sensor driver validates the data received from its sensor before forwarding it to the collector. If a violation is detected, the driver terminates itself so the supervisor can start another driver which in turn can connect to another physical sensor. The driver does however not attempt to correct the invalid values in any way.

4. ***Ensure carrying out critical functions at a specific time and in specific order***. Conflicts within task sequences can be resolved by terminating blocking processes which violate the order or a time constraint, as illustrated by the sensor calibration functionality. Thus lifelocks in calibration concurrency can be prevented – allowing only one sensor to calibrate at a time – and calibration of any sensor type is guaranteed within a given time-interval.

5. ***Unexpected failures have no influence or result in a safe state.*** Malfunctioning processes are immediately replaced by new ones, thus ensuring their functionality is not lost. Fatal function loss immediately results in system shutdown. For instance, the patient controlling system is robust with regard to sporadic process crashes as well as to the complete loss of one sensor data type.

E. Future work

The missing hard real-time abilities of Erlang pose a problem when it comes to time-critical safety applications. There are strategies to solve this issue, e.g. using external low-level libraries written in C/C++. These solutions have to be analyzed and developed further. For the applicability of LiC for safety critical systems, the underlying Erlang language features have to be evaluated against safety standards like IEC 61508-3. Research is also necessary on whether it is possible to apply LiC without Erlang. Analyzing the language features and corresponding counterparts in other languages or frameworks will provide the necessary information.

Impact of Feature Interaction on the Safety Analysis for Unmanned Avionics Product Lines

André L. de Oliveira[12], Rosana T. V. Braga[1], Paulo C. Masiero[1], Ibrahim Habli[2], Tim Kelly[2]
[1]Mathematics and Computer Science Institute, University of São Paulo, São Carlos-SP, Brazil
[2]Department of Computer Science, University of York, Deramore Lane, York, United Kingdom
{andre_luiz, rtvb, masiero}@icmc.usp.br, {ibrahim.habli, tim.kelly}@york.ac.uk

***Abstract*—Unmanned Avionics Systems (UAS) are real-time critical embedded systems that include high-integrity requirements. Most of these systems need to be certified before use, particularly in civil airspace. To reduce development cost, some UAS software is developed as part of a Software Product Line (SPL). A product-line comprises a reference architecture and a set of reusable core assets. New systems can be derived from the product-line architecture and core assets based on a predefined process that manages and controls permitted variations, based in part on product-line features defined in a feature model. However, many features are interdependent and hence complicate the analysis of all potential feature combinations for product-line systems. In this paper we discuss the impact of feature dependencies in the safety analysis of unmanned avionics SPLs and present a preliminary model-based solution for managing the impact of these dependencies.**

Keywords—unmanned avionics; product-lines; feature interaction;

I. INTRODUCTION

Unmanned aircraft systems are systems built to support aircrafts that do not require a human pilot [1]. Software Product Lines (SPL) consist of systems that share a common set of core requirements that differ according to a set of allowable variations [2]. SPL have been used to develop avionics software [3]. Safety-critical product-lines, such as avionics systems, include high-integrity safety requirements. For avionics SPLs to be used, it is necessary that both systems and aircraft be certified against pre-established guidance. Variability analysis and management is crucial for development of safety-critical SPLs, for which it should be considered in both product-line development and safety analysis. For example, safety case development [4] is an approach that has been used for documenting assurance arguments for safety-critical systems, such as avionics, in order to obtain certification credit. As with other product-line assets [2], product-line safety cases need to include mechanisms for managing the impact of variation [5]. Establishing a balance between safety assurance and reuse management is a challenging task in product-line safety analysis because a safety-critical SPL should satisfy its safety properties in all products derived from selecting and combining product-line features and assets.

II. RESEARCH PROBLEM

Feature interaction is defined as a feature or features affecting the behaviors of some other feature(s) [6]. This affects product-line safety analysis and development assets in both SPL domain engineering and application engineering. Thus, the following questions arise: a) how can product line safety/hazard analysis address feature interaction for avionics? b) How can assurance be provided that product-line assets are ready for reuse in several allowable configurations? Here we focus on two main challenges: 1) certification: certification authorities typically deal with *single* product certification and not with a product-line; and 2) feature interaction: dealing with the dependence relationships between product-line assets and how to provide assurance for the reuse of both product-line development and safety analysis assets. Feature interaction variation in product-line development assets and their operational environment (usage context) have a significant impact on safety analysis assets related to hazard identification, risk analysis, risk management (mitigation measures), risk monitoring, risk acceptance, and safety case argumentation. The addition of a feature into a safety-critical product-line can potentially lead to changes in many safety analysis assets, because it is necessary to consider and analyze the interaction of the new feature with other SPL features to perform safety analysis.

Variability management problems in avionics safety-critical SPLs relate to the complex traceability between functional dependencies (in aircraft functions) and product-line feature interaction and among product-line development, safety analysis, and safety assurance (safety cases) assets [4][5]. There are proposals for metamodels in the literature that address some of these problems, as the product-line Functional Failure Model [7] and the OMG Structured Assurance Case Metamodel [8], but there is little guidance, methods, or techniques that describe how to use such models together to manage such traceability. There is also no automated tool-support to use these models in order to analyze the traceability between product-line development, safety analysis, and safety argumentation assets.

III. PROPOSED SOLUTION

Our proposed solution to deal with feature interaction problems in avionics SPL development and safety assessment assets is based on the concept of 'problem-solution feature interaction' and a feature interaction mapping approach built based on this concept [6]. Problem-solution feature interaction is defined as an interaction between two or more architectural/implementation features (solution-space) that only arises based on one or more domain features (problem-space) from the feature model. The feature interaction mapping approach proposed by Sanen et al. [6] combines concepts of configuration knowledge and feature interaction, and the provisioning of automated tool support for complex mappings. In safety-critical SPLs, safety knowledge should be part of SPL configuration knowledge. Such knowledge covers SPL safety assets such as hazard logs, risk assessment, mitigation measures and argumentation data. So, in order to reuse knowledge about certain hazards and conditions in safety-critical SPLs we should incorporate 'safety knowledge' into 'configuration knowledge'.

We can abstract the 'problem-solution feature interaction' concept to address the traceability problem in safety-critical product lines. For example, avionics product-line feature models (domain models) map to the problem-space part of the concept, while Functional Dependency Models from avionics software can map to the solution-space part. We can also extend the concept of 'problem-solution feature interaction' to address safety (i.e.

variation interaction in safety assessment data) in safety-critical product lines. The reason for this is that there is also interaction between product-line development and safety assets. Thus, in the same way that the concept of 'problem-solution feature interaction' we can have interactions between one or more safety requirements (in the safety domain) that only arise in the presence of one or more feature (requirements/architectural) interactions in a specific usage context. To support modeling of feature interactions in SPLs, languages such as Feature-Oriented Requirements Modeling Language (FORML) [9] can be used. In this language, SPL modeling considers two viewpoints: the world problem, which comprises domain modeling using feature models to specify valid SPL combinations, and behavior model, to model feature interaction considering each SPL feature separately (feature module) using state-machines. FORML can be used to express feature interaction relationships in SPL feature and avionics functional models; and for expressing safety requirements interaction in safety-critical SPLs.

In order to address safety-critical product-line traceability for the UAS domain, we firstly propose a mapping between SPL feature interactions and avionics system function dependencies using merging metamodels, interfaces and parsing techniques. Model merging is a process of merging two source models 'M_A' and 'M_B', instances of 'MM_A' (feature interaction) and 'MM_B' (functional dependence) metamodels, into a target model 'M_C', which is an instance of 'MM_C' (merged) metamodel. We aim to use a merging language, such as Epsilon Merging Language (EML) [10], to build our proposed merged metamodel for feature interaction and avionics functional dependencies, within the Eclipse Modeling Framework (EMF). EMF will be used to provide automated tool support for traceability between product-line development, safety analysis, and safety argumentation assets.

We also propose other traceability merging metamodels to map core and variation points in product-line development (feature and context models), safety analysis (hazard identification, risk analysis, risk management), safety argumentation (safety cases) assets, manned and unmanned aircraft certification guidance, and product-line processes [2]. The merging metamodel for the safety case will be built based on the Goal Structuring Notation (GSN) [4][5] and the OMG SACM metamodel [8] and integrated with product-line processes [2]. From using our proposed merging metamodels, it will be possible to get traceability between a product-line feature associated with one specific usage context, and its correspondent safety analysis data, such as hazards related to the features in the assumed context, risk analysis data as risk severity and probability of occurrence, risk mitigation measures to be adopted, risk acceptability analysis, and safety argumentation (safety case models).

The presence of such traceability can contribute towards improvements in providing assurance of product-line features and feature interaction safety properties. This can be justified because the use of these metamodels can improve the management of product-line feature interaction safety requirements. We believe the use of such approach can facilitate the certification process of product-line configurations (through easier identification and management of dependencies) by reusing pre-certified safety analysis and safety argumentation data. The use of our merging metamodels can also contribute to reduce the complexity of adding new features and feature interactions to an existing product-line, due to the traceability between product-line interactions, safety analysis, and safety argumentation assets. To support and facilitate the use of our proposed metamodels, we are developing a UAS product-line development process and guidance to support the management of avionics software development, safety analysis and argumentation activities and their assets. Ongoing work involves developing tool support for both metamodels and the UAS development process, and validating the metamodels in real world case studies.

IV. RELATED WORK

Product line safety has been addressed in the literature, e.g. in Liu et al.[11], Habli et al. [7], and the MISSA Project [12]. Liu et al. [11] integrated SPL safety analysis with model-based development in a state-based modeling approach using two product-line safety analysis techniques: Software Failure Modes, Effects and Criticality Analysis, and Software Fault Tree Analysis. The MISSA Project [12] proposed a solution for assigning DALs for avionics systems developed from a set of models, by using Functional Dependency Models (FDM) and safety analysis tools. FDM is used for decomposing functions (features in an SPL) into sub-functions that correspond to classes or levels, or functional failure modes that impact the effects of a function failure condition. After all functional failure modes and all classes of functional performance are found, the decomposition is closed by allocating physical resources to implement the function. This data is processed by safety analysis tools to support the allocation of DALs to functions and their possible combinations. Habli et al. [7] proposed an SPL functional hazard model which is integrated with product-line context and domain (feature) models, and a model-based SPL hazard assessment approach aimed at integrating functional hazard assessment to product-line domain engineering and application engineering phases.

ACKNOWLEDGEMENTS

CNPq Brazilian research agency (grant 152693/2011-4).

REFERENCES

[1] J. P. Potocki de Montalk, "Computer software in civil aircraft', in: Proceedings of 6th annual conference on computer assurance, systems integrity, software safety and process security, 1991, pp. 10-16.

[2] P. Clements, L. Northrop, Software product lines: practices and patterns, Addison-Wesley Professional, 3a ed., 2002.

[3] F. Dordowsky, R. Bridges, H. Tschope, Implementing a software product line for a complex avionics system, In: 15th SPLC Conference, 2011, 241-250.

[4] T. Kelly, A systematic approach to safety case management, in: SAE world congress, Society for Automotive Engineers, 2003.

[5] I. Habli, T. Kelly, A safety case approach to assuring configurable architectures of safety-critical product lines, In: 1st ISARCS, Springer-Verlag Berlin, Heidelberg, 2010, 142-160.

[6] F. Sanen, E. Truyen, W. Joosen. 2009. Mapping problem-space to solution-space features: a feature interaction approach. In *Proc.*GPCE, ACM, 167-176.

[7] I. Habli, T. Kelly, R. Paige. Functional Hazard Assessment in Product-Lines: A Model-Based Approach. In *Model-Driven Product-Line Eenginering*, 2009.

[8] OMG, Structured Assurance Case Metamodel (SACM), available on-line: http://www.omg.org/spec/SACM.

[9] P., Shaker, J. M., Atlee, S. Wang, "A feature-oriented requirements modelling language," *Requirements Engineering Conference,* v. 151, n. 160, 24-28, 2012.

[10] D. S. Kolovos, R. F. Paige, F. A. C. Polack. Merging models with the epsilon merging language (EML). In *9th* MoDELS, Springer-Verlag, 215-229, 2006.

[11] J. Liu, J. Dehlinger, R. Lutz. Safety analysis of software product lines using state-based modeling, J. of Systems and Software, v80, n11, 1879-1892, 2007.

[12] MISSA Project, MISSA Project Final Report: Extract of the Publishable Summary, More Integrated Systems Safety Assessment, 2011.

A study on the reliability improvement factor of fault tolerant mechanisms

Jongwhoa Na, Dongwoo Lee
Department of Avionics and Electronics Engineering,
Korea Aerospace University,
Republic of Korea
{jwna, dongwoo1}@kau.ac.kr

***Abstract*—We present a study on the reliability improvement factor (RIF) to quantify the reliability of the various fault tolerant mechanisms at the system level. First, we find the system level failure rate using co-simulation models and statistical fault injection (StFi). We built co-simulation targets using SystemC simulation models of baseline single-core ARM7, dual-modular and triple-modular redundant ARM7 processors and Mibench embedded benchmark SW. Since the number of experiments in StFi is large, we utilized simulation kernel-modified simulated fault injection tool. Next, we calculated the RIF using the failure probability functions of the co-simulation targets. In this way, we were able to compare the reliability improvement of the fault tolerant mechanism at the system level.**

Keywords; statistical fault injection, reliability improvement factor, fault-tolerant processor, fault-tolerant mechanism

I. INTRODUCTION

In the safety-critical embedded systems (SCES) in aircrafts and automobiles, fault tolerant processors (FTP) became a major components. FTPs increase the reliability of the target using various types of redundancies. However, these redundancies also increase the cost of the target considerably. In order to manage the cost increase in SCES, we need a reliability index to quantify these redundancies. We may use MTTF as a reliability index for the target, which has components with sufficient usage history. However, because of the fast developing speed in VLSI/SoC technology, it is difficult to keep the usage history of the components of the modern SCES. This calls for a reliability index without usage history.

In this paper, we explain the application of the reliability improvement factor (RIF) as a reliability index of the effectiveness of the FT mechanism in the FTP. RIF is defined as the ratio of the probability of failure, F(t), of the non-redundant system to that of the redundant system [1,2]. For example, using ARM7 processor as a baseline processor, we may quantify the effectiveness of the TMR mechanism over DMR by finding the RIF_{TMR} of TMR ARM7 over baseline ARM7 and the RIF_{DMR} of DMR AMR7 over the same baseline.

We can calculate the F(t) of RIF by performing the fault injection experiments and finding the failure rate of the FTP and the baseline target. In order to make the failure rate legitimate, we use statistical fault injection (StFi) with confidence level and reasonable targets which can be a real SCES at the final stage or co-simulation target at the early stage of the development life cycle.

We report the case study of RIF for reliability index using StFi and co-simulation target. We built a co-simulation model of a SystemC hardware simulation model of baseline ARM7, DMR ARM7, TMR ARM7 processors and the cross-compiled executable files of the Mibench embedded benchmark suits [3]. For the statistical fault injection experiment, we calculate the required number of fault injections at a 95% confidence level for the given fault models and the SUT [4]. Because of the complexity of the SUT, the required number of fault injections is very large. Thus, the efficiency of the injection tool is important. In this regard, we use a novel simulated fault injection environment that uses a modified simulation kernel instead of saboteur or mutation technique. A detailed explanation of the kernel-modified simulated fault injection is explained elsewhere [5].

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors*).

II. FAULT TOLERANT PROCESSORS

For hardware, we designed a SystemC simulation model of the ARM7 processor that could execute about 40 instructions from the ARM7 architecture, as shown in Fig. 1.

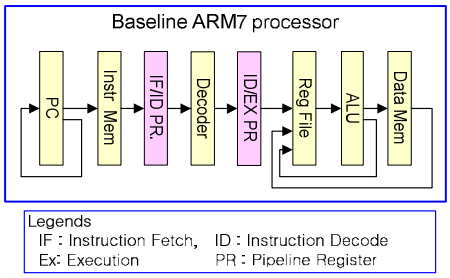


Fig. 1. ARM7 processor model

In order to make the cases of the qualitative comparisons of various FT mechanisms, we designed a SystemC simulation model for the DMR and TMR ARM7 processors. In the case of the DMR ARM7, we duplicated the data path with two ARM processors and added a simple fault recovery controller that could detect faults at the pipeline stages. In the design of the TMR ARM7 in Fig. 2, we implemented the micro-architectural redundancy by triplicating each module and adding a voter. The details of the DMR and TMR architectures can be found in many other studies [4].

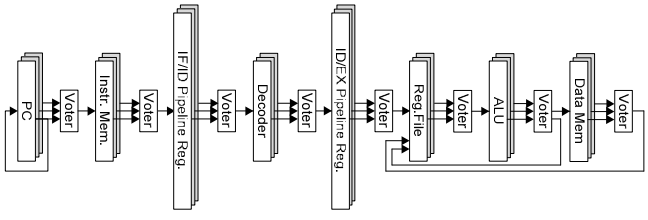


Fig. 2. Triple modular redundant ARM7 processor

III. STATISTICAL FAULT INJECTION EXPERIMENTS

We performed fault injection experiments using the co-simulation models using the three hardware models (baseline single ARM7, TMR ARM7, and DMR ARM7 processors) and the GSM code from a Mibench embedded benchmark suit, and the four fault models (permanent and transient stuck-at-1/0). For each of the experimental setups, we setup statistical fault injection experiments by calculating the required number of injections for the given confidence level. The results of fault injections are summarized in Table 1. In the case of baseline ARM7, the injection result should be one of the not-active, benign, or silent data corruption (SDC) state. In the case of DMR and TMR ARM7, we have two more states: recovered and detected unrecoverable error (DUE).

TABLE 1 The results of statistical fault injection campaign on co-simulation models and fault models

S/W		GSM benchmark (Mibench)			
H/W	Fault type & model / Failure type	Transient		Permanent	
		stuck-at-1	stuck-at-0	stuck-at-1	stuck-at-0
Base-line ARM	Non Active	28,468	71,191	8	31,244
	Benign	59,035	22,522	12,669	22,500
	Silent Data Corruption	12,497	6,287	87,323	46,256
DMR ARM	Non Active	29,046	71,314	125	32,950
	Benign	33,119	7,926	16,201	11,312
	Recovery	35,217	17,706	44,705	33,312
	Silent Data Corruption	519	769	75	1,008
	Detected Unrecoverable Error	2,099	2,285	38,894	21,418
TMR ARM	Non Active	47,554	74,237	229	5,867
	Benign	2,117	1,279	117	277
	Recovery	89,882	34,260	19,138	13,669
	Silent Data Corruption	447	224	516	187
	Detected Unrecoverable Error	0	0	0	0

Using Table 1, we calculate the failure rate by dividing the sum of the SDC and DUE failure rates by the total number of fault injections for the three types of processors. Using these failure rates and assuming steady state operating condition, we are able to calculate the reliability or failure distribution functions for each case of the baseline ARM, DMR ARM, and TMR ARM co-simulation targets. With the failure function, we can calculate the RIF of the DMR and TMR mechanism as follows:

$$RIF_{FTM} = (1- R_{baseline}(t))/(1-R_{FTM}(t))$$

We presents the RIF_{TMR} and RIF_{TMR} over baseline ARM7 processor in Fig. 3. Using the graph, we can compare the effectiveness of the TMR mechanism over DMR mechanism for given time in a quantitative manner. Initially, we can find the effectiveness of the TMR mechanism to be 50~70 times higher than that of the DMR mechanism. Also, we can find that the improvement of RIF_{TMR} over RIF_{DMR} decreases over time.

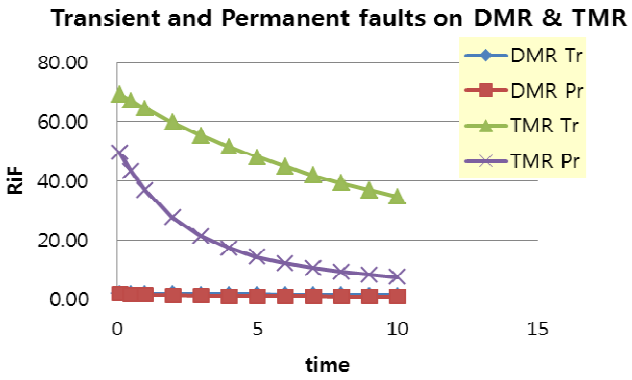


Fig. 3. DMR and TMR reliability improvement factors for the transient and permanent fault models

IV. CONCLUSION

We have reported on the reliability improvement factor (RIF) of the DMR and TMR mechanisms. Instead of using the static failure rates from the reliability block diagram, we utilized the dynamic failure rates using the co-simulation targets of SystemC hardware and Mibench benchmark software so that the RIF becomes more practical. The experimental results suggested that the TMR mechanism is initially more resilient than DMR. As such, we may compare the reliability or the cost-effectiveness of the FTM at the system level of various types of redundancy mechanisms.

Using these simulation study as a basis, we are planning to extend the experiments using other benchmark software and other FT mechanisms to investigate the applicability of the RIF as a quantitative reliability index of the fault tolerant mechanism for a group of embedded systems.

REFERENCES

[1] D.W. Lee and J.W. Na, "A Novel Simulation Fault Injection Method for Dependability Analysis," IEEE Design and Testing, vol. 26, no.6, pp. 50-61, Dec. 2009.

[2] Krol, Th., A Generalization of Fault-Tolerance Based on Masking, Ph.D. thesis, Eindhoven.

[3] Mei-Chen Hsueh, Timothy K. Tsai Ravishankar K. Iyer, "Fault Injection Techniques and Tools," IEEE Computer, April 1997.

[4] Shubu Mukherjee, "Architecture Design for Soft Errors," Morgan Kaufmann Publishers.

[5] Menkae Jeng and Howard Jay Siegel, "Implementation Approach and Reliability Estimation of Dynamic Redundancy Networks," Real-Time Systems Symposium, pp. 79-88, New Orleans, LA, 1986.