Diversity of Safety Arguments in the Validation of a Sounding Rocket Destruction System

M. A. D. Abdala; Institute of Aeronautics and Space; S. J. Campos, S. Paulo, Brazil

C. Lahoz; Institute of Aeronautics and Space; S. J. Campos, S. Paulo, Brazil

R. de Lemos, Ph.D.; Computing Laboratory, University of Kent at Canterbury, UK

Keywords: safety analysis, safety arguments, model checking, sounding rockets

Abstract

This work describes an approach for the validation of a software system responsible for the destruction of the sounding rocket VS-40X. The process of validation uses three different techniques ranging from the automatic state exploration to the laborious failure analysis. The purpose of the exercise was to obtain diverse arguments in the provision of evidence that the safety properties of the sounding rocket destruction system are always maintained. The software system is modeled using a co-operative architecture, which contains abstractions for modeling and analyzing the interactions between components. The safety analysis is performed using model checking, a technique that exhaustedly explores the state space to determine whether the system satisfies a safety property. The combination of co-operative architectures and model checking has shown effective when modeling and analyzing the interactive behavior between components. However, caution must be taken over the (false) confidence that can be obtained when employing solely model checking for the safety analysis. In order to compensate this deficiency we have to seek diverse sources of evidence to build trustworthy arguments about the safety of the system. The model checking was substantiated using laborious deductive and inductive analysis techniques.

Introduction

Safety analysis is a process that evaluates a critical system for determining whether the risk associated with the system is acceptable. The utilization of manually based safety analysis techniques, such as Fault Tree Analysis, Event Tree Analysis, or their variants, has shown not to be effective when performing the safety analysis of complex software intensive systems. Instead, alternative techniques, such as model checking, should be employed since they are tailored for analyzing highly complex systems.

Model checking automated tools might be effective in dealing with some of the inherent complexities of software based systems, for example dependencies of many failure modes of system components (ref. 1). However, for increasing the effectiveness of these tools support must be provided to enable the analyst to understand the rationale behind key safety-related decisions. Also, in order to reduce the risk of obtaining false confidence, the application of automated tools should be supported by a deductive and inductive analysis of the formal models subjected to automated analysis. In this paper, we present an approach in which traditional safety analysis techniques were used for validating the formal models required for the process of model checking. This work was performed in the context of the destruction system of the sounding rocket VS-40X, and consisted in replacing an existing remote destruction system for a self-destructing one. The latter would be essentially based on software, hence the need for providing evidence that its associated risk was acceptable. The evidence discussed in this paper is essentially qualitative rather than quantitative, and the criteria used for assessing the risk of the self-destruction system were obtained by the outcome of the safety analysis performed on the existing system.

The starting point of our approach was to model both systems in terms of a co-operative architecture, which is based on an architectural notation that is amiable in reducing the impact of change. That is, instead of building rather different models for an evolving system, we have chosen to employ modeling abstractions that are able to represent those aspects of a system that are more prone to changes. The utilization of the co-operative architecture for modeling the VS-40X destruction system has shown to provide the appropriate abstractions for performing safety analysis during the initial stages of the software design. This includes the identification of faults associated with the violation of safety specifications, and provision of evidence using model checking that the system behavior is safe (ref. 2). The confidence in using a co-operative architecture for modeling and analyzing a critical system was validated as described in reference 3. The aim of this paper is to devise an approach based on model checking and other safety analysis techniques that would allow to obtain diverse safety arguments for the provision of confidence that the risk of the new destruction system is acceptable. For that, laborious deductive and inductive analysis techniques were used in addition to model checking for performing the safety analysis of the co-operative architecture of the VS-40X destruction system. This safety evaluation was performed for two different destruction systems: remote destruction and self-destruction. The outcome of the safety analysis of the former was used as criteria to evaluate the safety of the latter. For the sake of brevity, we will present in this paper only the results of the evaluation process for the self-destruction system.

The rest of this paper is organized as follows. In section 2, the destruction system of VS-40X sounding rocket is described. In the section 3, we present the co-operative architecture of the self-destruction system. The results of the model checking of the co-operative architecture of the self-destruction system are presented in the section 4. In section 5, we present how deductive and inductive safety analysis techniques can be employed to provide diverse safety arguments. Finally, in the section 6, we summarize this work and provide some concluding remarks.

VS-40X Sounding Rocket Destruction System

The VS-40X Sounding Rocket: The VS-40X is a two stages sounding rocket that has a dual purpose within the Brazilian Space Program: in addition of performing scientific experiments, it will be also used as an experimental platform for the new equipment of the Brazilian Satellite Launcher (VLS). Currently, a safety operator destroys the rocket remotely, but the intention is to replace it with an automatic system for its self-destruction.

The flight of the VS-40X comprises the following phases: *pre-launching*, which starts with the beginning of the counting down until the instant in which there was the disconnection of the umbilical; *initial*, which is when the vehicle is in free evolution over the launch platform; *intermediate (I* and *II*), which comprises the interval initiating 5 seconds after the vehicle launching until the end of the propelled phase; and *final*, which starts with the ballistic flight until the end of the flight.

Mission and Safety Requirements of the VS-40X Sounding Rocket: Among the several mission requirements of the vehicle, in this work we consider the one that can influence, or to be influenced by the safety requirements, which is:

- The vehicle should not be destroyed under normal conditions during the flight.

The safety requirements of the VS-40X ensures the non-violation of the integrity of the environment of the vehicle, which is concerned with the potential damages to property and loss of lives when there is a system failure.

Depending on the phase of the flight and of the trajectory of the vehicle, we identify two accidents:

- An inadvertent destruction of the vehicle during pre-launching or the initial flight instants can cause damages to the launching installations, injuries or loss of lives;
- During the rest of the trajectory, the fall of debris, after a failure in the behavior of the vehicle, can cause damages to property, injuries or loss of lives.

There are two hazards (hazard_A and hazard_B, respectively) associated with the above two accidents:

- During the pre-launching and initial phases of the flight, there is a destruction of the vehicle;
- During the intermediate phases of the flight, the projection of the point of impact (PPI), of the vehicle's trajectory crosses the limit line of impact (LLI) and reaches the protected region (PR).

Based on the above hazards, we specify the following safety requirements for the destruction system of the VS-40X:

- The destruction of the vehicle should be disabled during the pre-launching and initial phases;
- The vehicle should be destroyed during the intermediate phases of the flight, once the trajectory of the vehicle violates the safety plan.

Destruction System of the VS-40X: Currently, the destruction system of the VS-40X is based essentially in the remote destruction: the safety operator is responsible for the activation of the destruction of the vehicle when there is a violation of the safety requirements.

However, the remote destruction system has some disadvantages: the information supplied by the radar can be imprecise depending on several problems related with tracing; the system is vulnerable and of difficult operation and maintenance; and pressures from the organization, or even emotional reasons, can make the safety operator to delay the remote destruction of the vehicle. Although the existing system is appropriate for simple launching vehicles, the idea is to employ the VS-40X as experimental platform for the self-destruction system for more sophisticated vehicles, such as, the satellite launcher.

Requirements for the Self-Destruction: The self-destruction system, without the aid of the safety operator, automatically destroys the vehicle whenever the safety requirements of the VS-40X are violated. The decision and command to destroy the vehicle are generated by the embedded protection system, which is enabled at the beginning of the intermediate phase of the flight. The components involved in the self-destruction are the protection system, the inertial reference system (IRS), and the trajectory calculation system.

During the intermediate phases of the flight, the data obtained by IRS are used to calculate the trajectory of the vehicle. This calculation allows the protection system to determine whether the vehicle should be destroyed when there is a violation of the flight safety plan.

Co-operative Architectural Style

Architectural structures for systems tend to abstract away from the details of a system, but assist in understanding broader system-level concerns (ref. 4). This is achieved by employing architectural styles that are appropriate for describing the software components, the interactions between these components, and the properties that regulate the composition of components. The co-operative architectural style adopts basic features of object-orientation, in which components are represented as classes and connectors as co-operative actions (CO actions). The instantiation of these abstractions are respectively, objects and co-operations. Objects are able to participate in several co-operations through the different roles that they are able to play while co-operations co-ordinate the interactions between the objects through the roles that objects play. The behavior of both objects and co-operations is described in terms of properties that have to be maintained for the system to provide the required services. The architectural elements of the co-operative style are classes as the basic components, and co-operative actions (CO actions) as the basic connectors. Co-operative actions (CO actions) were introduced as entities for modeling interactions between classes that characterize collaborative behavior. For describing the architecture of a software system, two different diagrams are employed: a *class diagram* describing the relationships between components, and a *CO action diagram* describing the relationships between connectors. These diagrams provide a compact representation of the software system, which can be completed with a more detailed textual description.

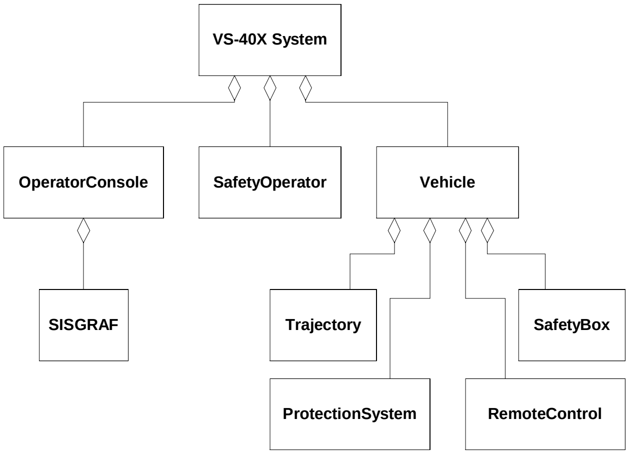


Figure 1 - Class diagram of the destruction system

The Co-operative Architecture of the VS-40X

The integrity of using the co-operative architecture for the modeling and analysis of critical systems was demonstrated in reference 3. That work presented an evaluation of the co-operative architecture of the remote destruction. A distinct feature of this approach compared with an object-oriented one is that focus is given to the behavior of the interactions between components, rather than on the components themselves. The evaluation of the approach was performed by comparing the minimal cut sets of the fault tree diagrams obtained from the natural language specification of the system and the corresponding co-operative object-oriented model.

In the following, we present the co-operative architecture of the self-destruction system of the VS-40X. In the class diagram of figure 1, the three basic components of the VS-40X System are the SafetyOperator, OperatorConsole, and Vehicle. In terms of the self-destruction system, the relevant components of the VS-40X are: SafetyBox that provides the protection mechanism to avoid the unintentional destruction of the vehicle during the pre-launching and initial phases of the flight; Trajectory that calculates the flight trajectory of the vehicle based on information provided by the Inertial Reference System (IRS); and the ProtectionSystem that establishes whether the flight safety plan has been violated.

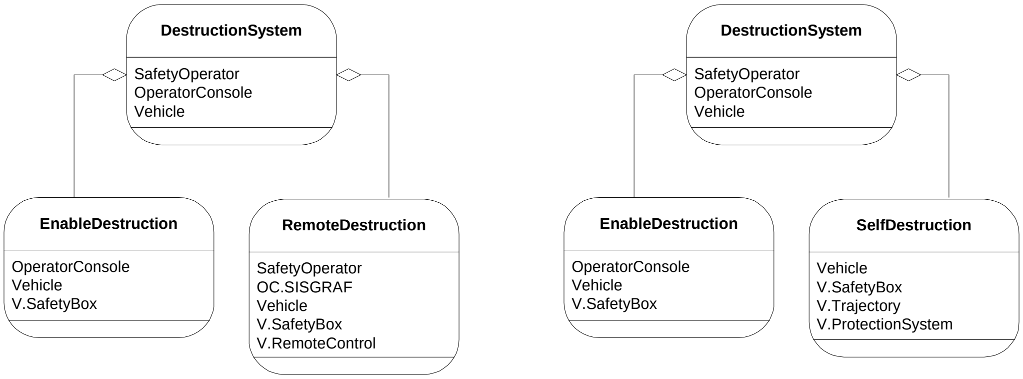
The diagram of figure 2 shows the CO actions associated with the remote and self-destruction systems of the VS-40X. The CO actions EnableDestruction and SeflDestruction are defined to represent, respectively, the enabling of the destruction and the actual self-destruction of the vehicle.

The co-operative action EnableDestruction, describes the interacting behavior of the components of the destruction system that enables the self-destruction of the vehicle. The normal behavior of EnableDestruction includes the pre-condition for the components of the system to enter the co-operative action: the safety operator enables the destruction of the system. The operation associated with this co-operative action: the safety box enables the destruction, 5 seconds after the rupture of the umbilical, once the operator enables the destruction of the system. The post-condition for the components of the system to leave the co-operative action: the destruction is inhibited by the safety box, or the vehicle is destroyed. The description of the failure behavior includes two conditions. A commission fault, when the destruction of the vehicle is enabled during the pre-launching and initial phases, and an omission fault, when the destruction is not enabled during the intermediate phases I and II.

The co-operative action SelfDestruction describes the behavior of the components for the self-destruction of the vehicle. The description of the normal behavior of the co-operative action includes the pre-condition to the system components enter in the co-operative action: enabling of the destruction by the SafetyBox. The invariant that establishes the condition that should hold during the execution of the co-operative action: the destruction is enabled by the SafetyBox. The operation associated with this co-operative action: the ProtectionSystem should destroy the vehicle when the vehicle leaves the protected region (PR), thus violating the safety plan. The post-condition for the components of the system to leave the co-operative action: the destruction of the vehicle is disabled or the vehicle is destroyed. There is a commission fault (commission1_SelfDestruction) related to hazard_A when the SelfDestruction is operational: Vehicle is either in pre-launching or initial phases of the flight, and the ProtectionSystem activates the destruction of the Vehicle. There is another commission fault (commission2_SelfDestruction) related to the violation of the missionRequirement when during the intermediate phases of the flight: the SelfDestruction is operational, the flight trajectory of the Vehicle is not outside the safety plan, but the ProtectionSystem activates the destruction. There is an omission fault (omission_selfDestruction) related to hazard_B when during the intermediate phases of the flight: the SelfDestruction is operational, the flight trajectory of the Vehicle is outside the safety plan, but the ProtectionSystem does not activate the destruction.

Model Checking

The safety analysis of the architectural representation should confirm that the combined co-operative behavior of the system CO actions is able to maintain the system safety. In the proposed approach, model checking, a formal verification technique based on state exploration



(i) Remote destruction

(ii) Self destruction

Figure 2 – CO action diagram of RemoteDestruction and SelfDestruction

has been employed for obtaining evidence that system safety is not violated.

While the normal behavior of the system is expressed as state transitions, the failure behavior is specified as reachability and invariance properties corresponding to failure behaviors of the CO actions. Given a state transition system and a property, model checking algorithms exhaustedly explore the state space to determine whether the system satisfies the property. The result is either a claim that the property is true or a counter-example in terms of a sequence of states that falsifies a property.

The model checker employed for performing the verification of the CO action behavioral specification is UPPAAL, an automated tool for the analysis of real-time systems (ref. 5). The model checker in UPPAAL can check reachability and invariance properties of Boolean combination of automata locations, and clocks and integers constraints. E<>ϕ expresses the possibility of reaching a state satisfying ϕ, which might be of the form "ϕ is guaranteed to hold within time t" and may be used to verify that an expected situation occurs within a specified time bound. Dually, A[]ϕ expresses invariance of ϕ, which might be of the "ϕ is always true" and may be used to verify that certain situations never occur. The safety properties to be confirmed are obtained from the specification of failure behavior of a CO action.

The system context for conducting the safety analysis of the self-destruction system consists of the automata presented in figure 3: EnableDestruction and SelfDestruction. In the EnableDestruction automaton, the destruction of the vehicle will be enabled after the safety operator enables the system destruction (v_sb_safeDest) and the safety box also enables the destruction, five seconds after the rupture of the umbilical (v_sb_enDest). In the selfDestruction automaton, if the destruction is enabled (v_sb_enDest) and the component trajectory detects that the flight trajectory of the vehicle is outside the safety plan (v_outsideSP), the protection system will activate the destruction of the vehicle (v_ps_actDest). Besides these two automata, other automata were constructed for representing the flight phases, the safety operator and the safety plan.

For the analysis of the safety properties of the destruction system, the first step was to check using UPPAAL queries whether the normal behavior of SelfDestruction would not violate its safe behavior. The next step was to check whether the combined behavior of CO actions EnableDestruction and SelfDestruction is able to maintain the safety of the VS-40X, as we have already checked for the RemoteDestruction system. After making the co-operative model of the self-destruction system, we proceed with the verification of the safety properties using the UPPAAL model checker. We identified the following properties that would be satisfied during the process of model checking:

- A[] not ((FP.FP0 or FP.FP1 or FP.FP2) and v_destroyed==1)
- A[] not ((FP.FP0 or FP.FP1 or FP.FP2) and v_ps_actDest==1)
- A[] not ((FP.FP0 or FP.FP1 or FP.FP2) and ED.v_sb_enDest==1)
- E<> not ((FP.FP3 or FP.FP4) and (v_outsideSP==1 and v_destroyed==0)
- E<> not ((FP.FP3 or FP.FP4) and (v_tr_outsideSP==1 and v_ps_actDest==0)
- E<> not ((FP.FP3 or FP.FP4) and ED.v_sb_enDest==0)

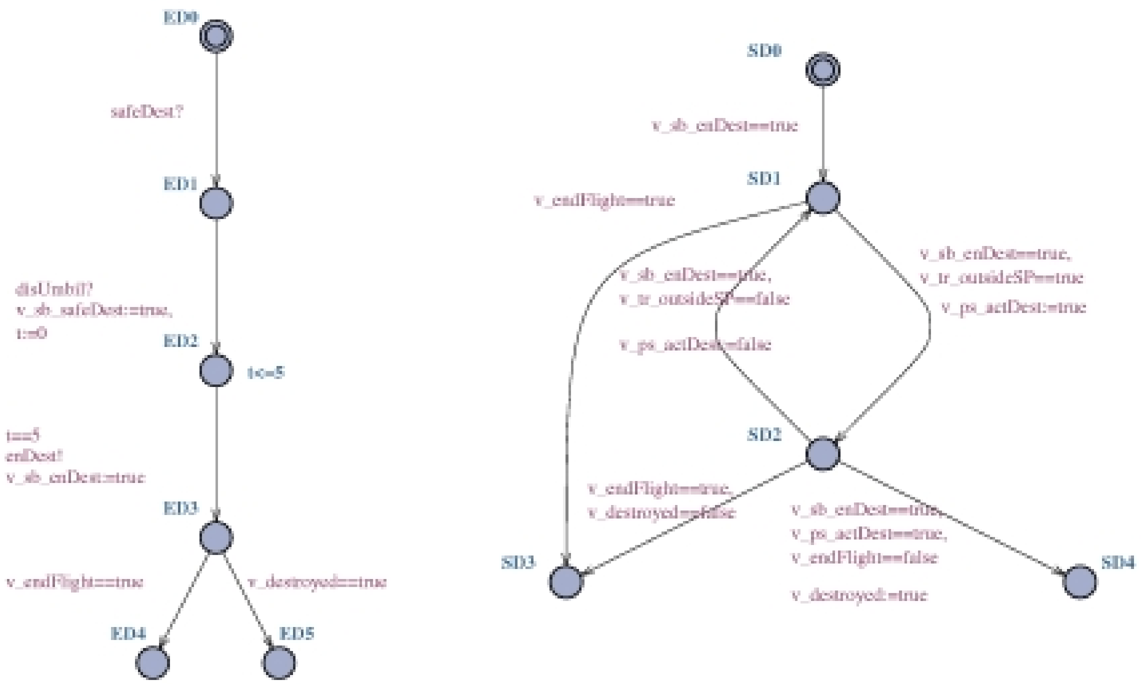


Figure 3 - Hybrid automata of EnableDestruction and SelfDestruction

- A[] not (FP.FP3 or FP.FP4) and (v_tr_outsideSP==0 and v_ps_actDest==1).

Using the UPPAAL model checker, we have confirmed that the invariance and reachability properties associated with all CO actions were satisfied, that is, every automata model of the CO actions is able to maintain its associated safety behavior. Also we checked that the System as a whole was able to maintain its safe behavior.

Diversity in the Safety Arguments

A typical approach for the application of model checking to safety analysis is: the property model to represent the safety property that the system has to satisfy, usually associated with the negation of the system hazard, and the operational model to represent the system being designed, including the possible failures of the components of the system. For model checking to be effective as a safety analysis technique, it should support risk reduction and provide evidence for safety (ref. 2). In terms of risk reduction, model checking can identify the possible causes for the violation of the properties associated with the model. Once these causes are identified the model can be modified to eliminate or mitigate the risk (if both the property and model cannot be modified then risk remains unchanged). In terms of evidence, model checking can show that despite failures in the components of the system, the safety properties of the system are not affected (or if affected the risk associated with the failures is acceptable). As with any modeling technique, the confidence that can be attributed to the results obtained from model checking is dependent on the accuracy of the models, hence property and operational models should be validated to confirm they are accurate representations of the actual system. Although it is relatively easy to check whether the operational model satisfies the specified properties, there are several error sources in the process of modeling. For example, either the property or operational models might have a mis-representation (inappropriate parameter which defines states/transitions of the automata, or flawed initial conditions) that allows a property to be confirmed for the model despite it being inappropriate for the real system. In particular, an analogy can be made with testing when applying model checking as a safety analysis technique: model checking is able to confirm the presence of faults in the model, but not their absence. Moreover, while testing is able to probe the actual product being developed, model checking is only restricted to probe a representation of the actual product. Hence, additional assurance should be provided that either the model being checked is an accurate representation of the system, or that all the exposed inaccuracies between the model and the actual system do not impact system safety.

In this paper we claim that model checking is an effective technique when used in addition with

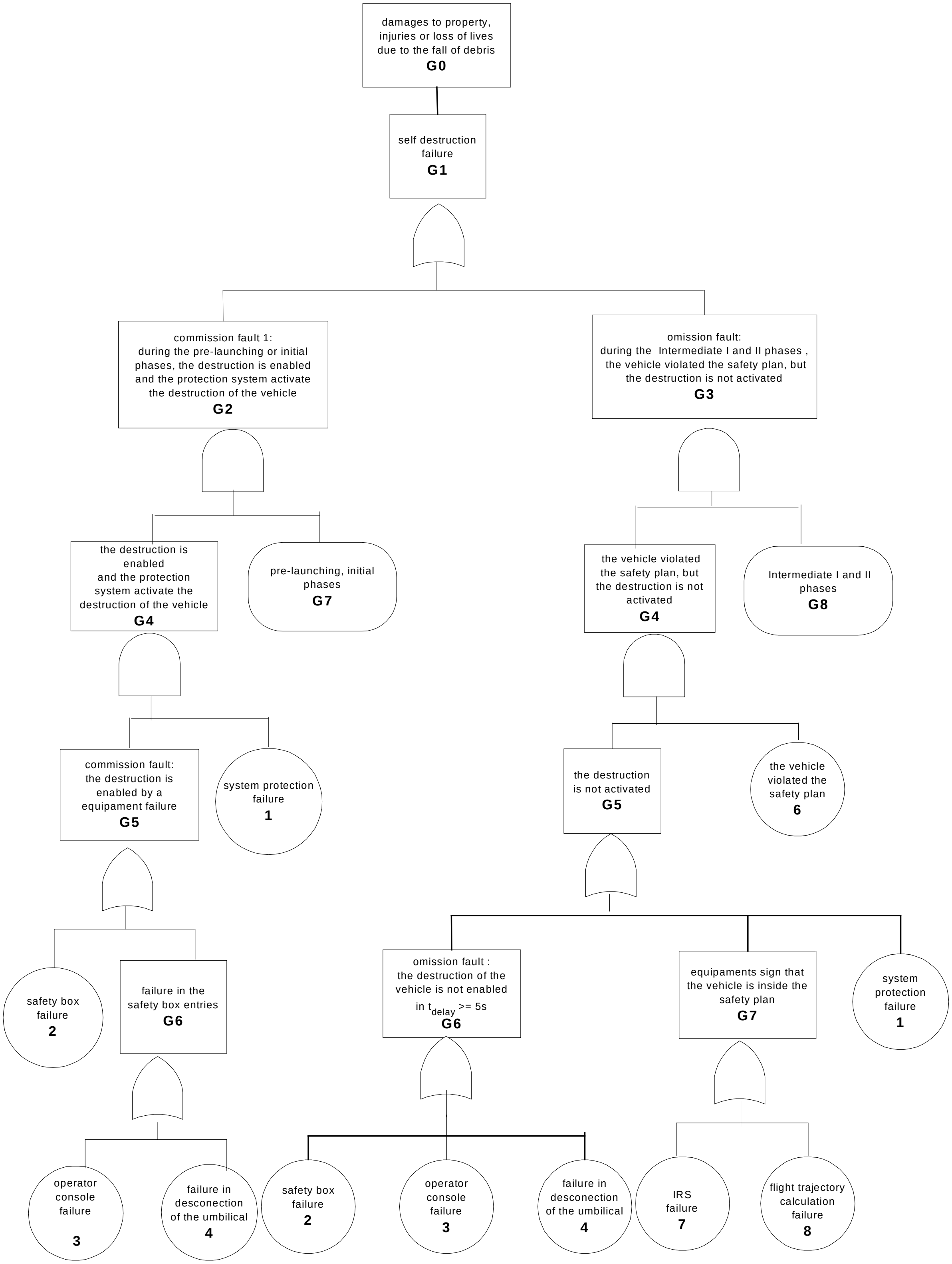

damages to property, injuries or loss of lives due to the fall of debris
G0
self destruction failure
G1
commission fault 1: during the pre-launching or initial phases, the destruction is enabled and the protection system activate the destruction of the vehicle
G2
omission fault: during the Intermediate I and II phases , the vehicle violated the safety plan, but the destruction is not activated
G3
the destruction is enabled and the protection system activate the destruction of the vehicle
G4
pre-launching, initial phases
G7
the vehicle violated the safety plan, but the destruction is not activated
G4
Intermediate I and II phases
G8
commission fault: the destruction is enabled by a equipament failure
G5
system protection failure
1
the destruction is not activated
G5
the vehicle violated the safety plan
6
safety box failure
2
failure in the safety box entries
G6
omission fault : the destruction of the vehicle is not enabled in t_{delay} >= 5s
G6
equipaments sign that the vehicle is inside the safety plan
G7
system protection failure
1
operator console failure
3
failure in desconection of the umbilical
4
safety box failure
2
operator console failure
3
failure in desconection of the umbilical
4
IRS failure
7
flight trajectory calculation failure
8

Figure 4 - Fault Tree of SelfDestruction

other safety analysis techniques for conducting the safety analysis of VS-40X destruction system, because these techniques can provide means for validating the operational and property models of the actual system. In the following, we proceed to perform the safety analysis of the destruction system of the VS-40X, in particular the SelfDestruction, using deductive and inductive safety analysis techniques. The exercise consisted of obtaining the fault and event trees of the co-operative architecture of the SelfDestruction. The outcome of this analysis is then used to substantiate the analysis performed using model checking by validating its operational and property models, according with the following guidelines: check whether the component failures related with the primary events are capture in the extended timed automata representation, check whether the invariance and reachability formula captures all the expected failure behaviors of the system, and check whether the automata traces of the model checking counter-example that falsifies a safety property are equivalent to the sequence of events in a event tree.

Fault-Tree Analysis (FTA): The goal of this technique is to determine the causes for the occurrence of an undesirable event, which in the safety analysis is related to a hazard. This technique, through a deductive approach, starts from a system hazard and searches backward the faults of the system that could cause this hazard.

In the FTA of the VS-40X destruction system we identified the following top events and their respective primary events (figure 4):

- During the pre-launching or initial phases, the destruction is enabled AND the protection system activates the destruction of the vehicle (commission fault 1): system protection failure AND (safety box failure OR operator console failure OR failure in disconnection of the umbilical;
- During the Intermediate I and II phases, the vehicle violates the safety plan, but the destruction is not activated (omission fault 2): system protection failure OR IRS failure OR flight trajectory calculation failure OR safety box failure OR operator console failure OR failure in disconnection of the umbilical;
- During the Intermediate I and II phases, the vehicle is inside the safety plan, but the destruction is activated (commission fault 2): system protection failure OR IRS failure OR flight trajectory calculation failure.

Initially, we have checked whether the primary events were captured by the automata model, and the top events by the property model. For example, the third fault listed above was represented as a condition that cannot be reached by the system, if safety has to be maintained. The fault tree allows representing the combination of faults that could lead the system to enter a hazard state. This process helps the safety analyst to understand the causes of possible failures, thus allowing the models to be improved.

Event-Tree Analysis (ETA): This technique uses forward search to identify the various possible outcomes of a given initiating event, determining all sequences of events that could follow it. This technique was applied to the self-destruction system of VS-40X (figure 5) to identify the sequence of events that could lead to the violation of a mission or safety requirements, as previously stated:

- Enable destruction fault - protection system activates the destruction AND the trajectory OR IRS fails;
- Vehicle is not destroyed when it is outside the safety plan - IRS fails OR trajectory fails OR protection system fails (by not destroying the vehicle);
- Vehicle is destroyed when it is inside the safety plan - IRS fails OR trajectory fails OR protection system fails (by destroying the vehicle).

With the event tree analysis, we have checked whether the automata traces of the counter-example that falsifies a safety property are equivalent to the sequence of events in an event tree. That is, we evaluated the sequence of events that leads to a system failure, identified as safety property that would not be satisfied in the model checking, and then check if the transitions between the states of the automata model that falsifies the safety property are equivalent to this sequence.

After conducting the cross checking of the models according to the guidelines previously identified, we have concluded that models used by model checking, and their associated assumptions, were an accurate representation of the SelfDestruction system. The evidence that the automated destruction system (SelfDestruction) was able to maintain the safety and mission requirements of the sounding rocket VS-40X was based on a set diverse safety arguments obtained from the different analysis performed.

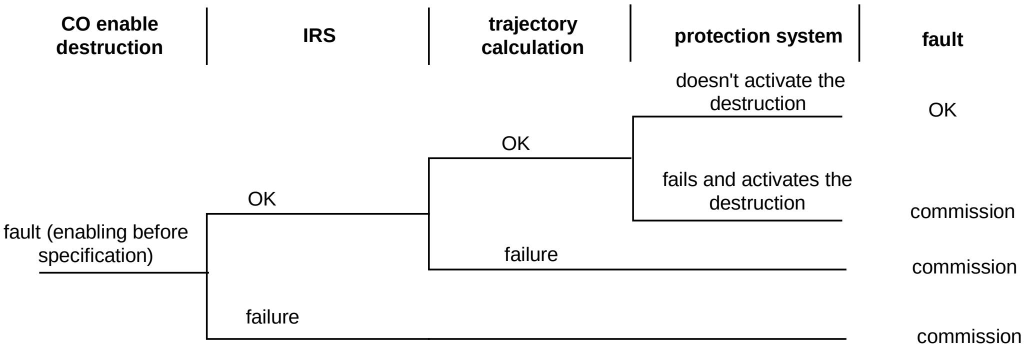


Figure 5 - Event Tree of CO Enable Destruction

Conclusions

This paper has presented the use of different techniques for the purpose of obtaining diverse safety arguments in the provision of evidence that the safety of the system cannot be violated. Instead of relying solely on laborious deductive or inductive safety analysis techniques, we argued that alternative techniques, such as model checking, should be used because they are tailored for analyzing highly complex systems. However, the novelty in using this technique in safety analysis and the non-existence of a well-trusted process for its application requires additional evidence to be produced in order to avoid the risk of obtaining false confidence.

The approach being proposed employs both model checking and more traditional safety analysis techniques. However, instead of relying on the former for obtaining assurance about the integrity of the system, we have used fault and event tress to complement the outcome of the safety analysis based model checking. The outcome of combining model checking and fault tree analysis allows to check whether the component failures related with the primary events are capture in the extended timed automata representation, and whether the reachability formula captures all the expected failure behaviors of the system. The outcome of combining model checking and event tree analysis allows checking whether the automata traces of the counter-example that falsifies a safety property are equivalent to the sequence of events in a event tree. The feasibility of the proposed approach was demonstrated through the specification and verification of the destruction system of a sounding rocket. The combine use of model checking, fault and event tree analysis has provided qualitative evidence that safety properties of the destruction system are maintained.

Acknowledgements

The authors would like to acknowledge the financial and logistic support of CAPES, of the British Council, and of the Institute of Aeronautics and Space, IAE.

References

1. A. L. Turk, S. T. Probst, and G. J. Powers. Verification of Real Time Chemical Processing Systems. *Proceedings of the International Workshop on Hybrid and Real-Time Systems*. Lecture Notes in Computer Science 1201. Ed. O. Maler. Grenoble, France. March 1997. pp. 257-272.

2. R. de Lemos, A. Saeed. Validating Formal Verification using Safety Analysis Techniques. *Proceedings of the 18th International Conference on Computer Safety, Reliability and Security (SAFECOMP'99)*. Toulouse, France. September, 1999. pp. 58-66.

3. C. Lahoz, M. Abdala, C. A. T. Moura, R. de Lemos. Evaluation of Co-operative Actions in the Safety Analysis of the Destruction System of the Sounding Rocket VS-40X. *Proc. of the Symposium on Safety and Security of Information Systems*. São José dos Campos, Brazil. October 2000. pp. 49-58. (In Portuguese)

4. M. Shaw, D. Garlan. Software Architectures: Perspectives on an Emerging Discipline. Prentice-Hall, Inc. Upper Saddle River, NJ. 1996.

5. K. G. Larsen, P. Pettersson, W. Yi. UPPAAL in a Nutshell. *International Journal on Software*

Tools for Technology Transfer 1(1–2). October 1997. pp. 134–152.

Biography

M. A. D. Abdala; Technologist, Institute of Aeronautics and Space, Pça Mal Eduardo Gomes, 50, S. J. Campos, S. Paulo, Brazil, telephone – (55 12) 3474968, facsimile – (+55 12) 3475019, e-mail – martha@iae.cta.br.

She is a system analyst, who has been working for the last sixteen years for the Brazilian government Institute of Aeronautics and Space (IAE/CTA). She has been involved in the development of the on-board software for the Satellite Launch Vehicle (VLS).

C. Lahoz; Technologist, Institute of Aeronautics and Space, Pça Mal Eduardo Gomes, 50, S. J. Campos, S. Paulo, Brazil, telephone – (+ 55 12) 3474901, facsimile – (+55 12) 3475019, e-mail – lahoz@iae.cta.br.

He is a system analyst, who has been working for the last seventeen years for the Brazilian government Institute of Aeronautics and Space (IAE/CTA). He has been involved in the development of the on-board software for the Satellite Launch Vehicle (VLS).

R. de Lemos, Ph.D.; Lecturer, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, UK, telephone – (+44 1227) 823628, facsimile – (+44 1227) 762811, e-mail – r.delemos@ukc.ac.uk

He is a lecturer at the University of Kent at Canterbury since 1999, and before that he was a senior research associate for several years at the Centre for Software Reliability at the University of Newcastle upon Tyne. His area of interest is software development for safety-critical systems, in particular, application of formal methods, validation of formal models and integration of requirements analysis and safety analysis.