# COMPASS3 – Roadmap

COMPASS Consortium

December 16, 2016

# Contents

**Abstract**

COMPASS is a toolset for model based verification, safety and performability analysis of complex aerospace systems. It has been developed mostly under funding of the European Space Agency (ESA), in response to the need of a more formal and comprehensive approach to the problem of system-software co-engineering.

It this document we discuss the COMPASS roadmap, that is, we identify the COMPASS objectives, the current status and the future steps needed to reach the objectives.

# Chapter 1

# Introduction

COMPASS is a toolset for the evaluation of system-level correctness, safety, dependability and performability of on-board computer-based aerospace systems. It supports a comprehensive process for system-software co-engineering, by covering requirements validation, functional correctness, safety and dependability analysis, performability analysis, fault detection, identification and recovery, and contract-based design.

The COMPASS toolset has been developed since 2008, with funding of the European Space Agency, by Fondazione Bruno Kessler (FBK), Trento, and RWTH Aachen University, in several projects such as COMPASS [29], AUTOGEF [7], FAME [33], HASDEL [37], D-MILS [31], CITADEL [28] and CATSY [21]. COMPASS is the original ESA-funded project where the toolset was first developed. AUTOGEF, FAME, HASDEL and CATSY are follow-up ESA-funded projects focusing on different extensions, namely the FDIR development and synthesis, the modeling and verification of failure propagation information, the extension of RAMS (Reliability, Availability, Maintainability and Safety) to the specific (real-time) needs of launcher systems, and finally the definition of a Catalogue of System and Software Properties (CSSP), to be derived from a taxonomy of requirements. The D-MILS project and its follow-up CITADEL, on the other hand, are two EU-funded project that focus on the compositional system construction and assurance for the design and certification of distributed systems, and its extension to adaptive systems.

More recently, the COMPASS3 project has addressed the consolidation of the COMPASS toolset and the release of a new version: COMPASS 3.0, which is the starting point for the discussion of the COMPASS roadmap in the present document. Fig. 1.1 shows a historical perspective of the COMPASS toolset versions, where COMPASS release 2.2 and 2.3 were delivered in the COMPASS project, COMPASS release 3.0 in the COMPASS3 project, and the other releases were part of their respective projects. Here, arrows represent dependencies/functionality inclusion between different projects/releases.

In this document, we develop the COMPASS roadmap. More concretely, we discuss the COMPASS objectives, we analyze the current status of COMPASS, and discuss the future steps to reach the objectives. The document is structured as follows:

- In Chapter 2 we identify the COMPASS objectives.

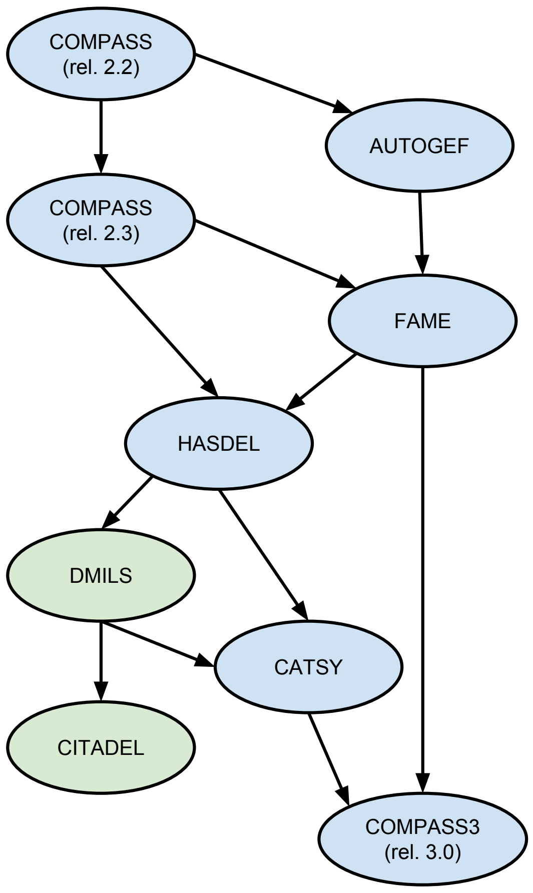

Figure 1.1: COMPASS toolset versions.

- In Chapter 3 we describe the current status of COMPASS.
- In Chapter 4 we discuss the future of COMPASS.
- In Chapter 5 we present the action planning.

# Chapter 2

# COMPASS Objectives

In this chapter we identify the COMPASS objectives. We first discuss toolset, process and research objectives, then we identify our expectations regarding the impact of COMPASS on the community at large, and finally we discuss the integration with other ongoing activities at ESA.

## 2.1 Toolset

We identify the following objectives related to the toolset development, and distribution.

**T1** Enhance the usability of COMPASS. This includes enabling the use of different input languages, supporting the integration with existing tool chains, and improving and enriching the available functionality.

**T2** Bring the COMPASS toolset to higher levels of technology readiness. This includes making the toolset more robust and portable, improving the user interfaces, and enhancing the scalability.

**T3** Increase the accessibility of COMPASS. This includes licensing considerations (compare Section 3.1.7).

## 2.2 Process

We identify the following process-related objectives. We collect both objectives that are relevant to the development process of the toolset itself, and the use of the toolset in the system design process.

**P1** Enhance the development infrastructure of the COMPASS toolset, in particular build upon continuous integration technologies, in order to make the development of COMPASS more effective.

**P2** Make COMPASS amenable to integration with existing standards, such as the ECSS standard used in ESA.

**P3** Make COMPASS amenable to supporting certification activities.

## 2.3 Research

We identify the following research objectives.

**R1** Demonstrate the applicability of formal methods technologies for designing (industrial) systems of realistic size.

**R2** Explore research advances that can improve the applicability and scalability of formal methods technologies, such as new algorithms and new verification paradigms, or tuning existing routines and engines.

**R3** Increase visibility of the research activity underlying the COMPASS toolset. This includes publishing the results in international conferences and journals, presenting the results in tutorials, courses, PhD schools, and making students interested in the field of formal methods applied to system design.

## 2.4 Community

A major objective of COMPASS is to increase the impact on the community at large, including the research community and the industrial users. We identify the following main goals.

**C1** Improve visibility of the COMPASS toolset.

**C2** Improve market penetration, including commercial exploitation, of the COMPASS toolset.

**C3** Increase industrial usage of the COMPASS toolset.

In general, we think that, in order to push industrial usage, a policy based on small steps is more likely to success than proposing a disruptive change. Integration into current practices must be supported in order to enable the uptake of COMPASS.

## 2.5 Integration with ESA Initiatives

Another important goal of COMPASS is the integration of the COMPASS toolset into other ongoing ESA-related activities. Such integration will help in positioning COMPASS within the overall design process of (aerospace) systems, and increase the potential for its usage in the community.

We identify the following COMPASS-related initiatives: TASTE, OSRA, CSSP.

**TASTE** TASTE is a development environment dedicated to embedded, real-time systems and was created under the initiative of the ESA back in 2008, after the completion of a FP6 project called ASSERT. TASTE can be used to design small to medium-size systems; it relies on formal languages and is based on the idea of building "correct by construction" software. The modeling language provides a mixture of languages used for different purposes: AADL for the system-level view of the architecture and ASN.1 for data abstraction and implementation, while different languages can be employed for behavior specification, e.g. SDL. Finally, native support for property specification and verification is lacking.

**OSRA** The Space AVionics Open Interface aRchitecture (SAVOIR) is an ESA initiative to federate the space avionics community and to work together in order to improve the way that the European space community builds avionics sub-systems. In particular, the goal of the subgroup SAVOIR-FAIRE (SAVOIR Fair Architecture and Interface Reference Elaboration) is to achieve the definition of an On-board Software Reference Architecture (OSRA). OSRA comprises three architectural layers: the component layer, the interaction layer, and the execution platform. The Space Component Model (SCM) is a reference implementation for the OSRA component layer.

**CSSP** CSSP is an ESA initiative for the definition of a Catalogue of System and Software Properties, to be derived from a taxonomy of requirements and to support the design of space systems with formal properties. ESA funded two parallel studies to develop such a catalogue, one called CATSY and lead by SSF with RWTH and FBK as subcontractors, the other lead by University of Thessaloniki with EPFL and Thales as subcontractors.

We identify the following objectives.

**I1** Integrate COMPASS and TASTE, in order to bridge the gap between architectural level design and system implementation and deployment.

**I2** Make models used in COMPASS and TASTE compliant with the On-board Software Reference Architecture.

**I3** Define an FDIR reference architecture, by inspiration to OSRA, and make COMPASS the reference tool for such architecture.

# Chapter 3

# Current Status of COMPASS

In this chapter, we discuss the current status of COMPASS, and in particular we identify strengths and limitations that may affect the impact of COMPASS and its introduction in industrial practice. Chapter 4 will then discuss the strategy and future steps needed to address the identified limitations.

## 3.1 Toolset

In this section we discuss issues related to the current status of development and distribution of the COMPASS toolset.

### 3.1.1 SLIM Language

The SLIM language was originally designed as a dialect of AADL Version 1 [3], to meet the needs of the European space industry. The base language is mainly focused on the architectural organization of a system under nominal and degraded modes of operation. Our goal was to extend the language beyond AADL focus, and define the architecture of a system by also analyzing its dynamic behavior, namely both its nominal and degraded modes of operation and their interweaving. Moreover, we addressed modeling of partial observability, timed and hybrid behavior, and probabilistic aspects, such as random faults, repairs, and stochastic timing.

Successor activities of the original COMPASS project, in particular those listed in Chapter 1, have created the need for adding further language features and for changes in the semantics. This includes, among others, observability and non-blocking attributes for ports, passive port connections (which do not influence observable behavior), timing-related constructs (urgency and delays), new security-related concepts (e.g., keys data types and encryption operations) and annotations (akin to AADL annexes).

This has incurred several syntactic and semantic issues regarding the interpretation and analysis of specifications by different implementations of the COMPASS toolset. These design decisions also implied that the SLIM language was incompatible with AADL, though strongly resembling it. For better adoption, as indicated by the community, it made sense to bring SLIM and the AADL closer together as existing AADL-based workflows can then be employed. This also would allow COMPASS to focus on core functionality, while language oriented tooling (such as the OSATE development environment [44]) is being available from the AADL community.

One major goal of releasing the new modeling language version SLIM 3.0 was therefore to provide a clear and unified definition of the syntax and semantics of the SLIM language. This was accomplished by the elimination of useless constructs and the inclusion of selected constructs from previous projects such as event-data ports, parameters, user-defined types, tuples, functions, time units, urgent locations, and urgent transitions. At the same time, the semantics was consolidated by resolving existing issues such as the interpretation of communication along event port connections, the encoding of integers as words and the presence of non-deterministic inputs and of non-deterministic unconnected ports.

COMPASS3 has also improved the alignment between SLIM and AADL Version 2 [4], bringing the syntax of both languages together. This goal was essentially achieved by employing AADL's "properties" mechanism to extend its syntax by SLIM-specific features like data types, special-purpose attributes of architectural elements such as observability properties of events or default values of data ports, and contracts for formalizing requirements on a component's behavior.

### 3.1.2 Modeling

Currently, the COMPASS toolset supports SLIM as input language. SLIM, as described in Section 3.1.1, was defined to be a variant of AADL, which has a consolidated user base in the academic and also industrial community. However, there are several other languages that are being used in the current industrial practice, for which COMPASS currently provides no support.

In general, the cost of modeling is a real barrier for adoption of a verification toolset in the industry. Most industrial companies are not willing to adopt languages that differ from those being used in their internal processes. Moreover, the COMPASS limitation to a single modeling language prevents re-use of existing models. For these reasons, there is a need to address this limitation with high priority. A direction to be investigated is enhancing COMPASS with a front-end that can take different input languages and transparently convert them into the internal format used by the verification engines. The internal format, in this view, can be the common ground (both syntactically and semantically) to link the different input languages. The mix-and-match integration of models written in different languages, on the other hand, is perceived to be of lower priority by the industrial community.

### 3.1.3 Integration with Design Environments

Similarly as for modeling languages, the integration of the COMPASS toolset within consolidated modeling and design environments used in the industry could also positively affect the adoption of COMPASS in the industry. Currently, there is no such integration.

### 3.1.4 Functionalities

COMPASS incorporates a wide range of verification functionalities that cover many aspects of system design, from modeling to verification and validation. These functionalities include requirements validation, functional correctness, safety and dependability analysis, performability analysis, and fault detection, identification and recovery. Moreover, COMPASS 3.0 integrates the contract-based design functionality, which enables a component-oriented design based on the specification of the components' behavior (via assumptions and guarantees) and iterative refinement.

The outcome of the 2015 COMPASS Workshop suggested the need for integrating additional functionalities. In particular, the following ones have been identified: model validation (i.e., ensuring the quality of a formal model with respect to what the user has in mind), model-to-model comparison (useful to manage change and evolution, and for model re-use) and model documentation (i.e., generation of documentation and artifacts that can be useful for the comprehension of models). The 2016 MBSSE Workshop also suggested to take into account the issue of change management, in particular to deal with model change and evolution,and traceability between models and analysis results.

### 3.1.5 Scalability

An important aspect that may impact the level of maturity, and technology readiness level, of the COMPASS toolset is the scalability of its functionalities. The COMPASS toolset is the outcome of a significant research effort, and incorporates state-of-the-art edge verification technologies. COMPASS 3.0 incorporates further verification techniques such as advanced model checking techniques, new engines and options, used for functional verification and safety analysis, that aim at improving the scalability of these activities [15, 17, 35, 27]. Moreover, the use of contract-based design techniques, that are amenable to compositional verification, is a powerful approach to dominate the complexity of verification.

However, it is well known that some analyses are inherently highly complex, thus they may hinder the application of COMPASS to the most complex models. For these reason, further means of mitigation must be investigated. This includes further tuning of the verification engines, and the use of techniques that trade between the precision of results and the analysis effort.

### 3.1.6 Case Studies

The applicability of COMPASS has been demonstrated in several case studies in the space domain and in other domains, see [19]. Moreover, within the COMPASS3 project, existing examples coming from different predecessor projects have been systematized and extended, in order to be representative of all the functionalities of the toolset. Hence COMPASS 3.0 now includes a more comprehensive suite of examples.

However, COMPASS is still lacking some bigger case studies, that may be representative of realistic industrial systems. It is fundamental to have access to such case studies, in order to improve the maturity and technology readiness level of the toolset. Currently, such case studies either do not exist, or they cannot be disclosed to the community since they are protected by confidentiality agreements.

### 3.1.7 Software and Licenses

The COMPASS3 initiative has consolidated the COMPASS toolset in terms of syntax and semantics, software architecture and software quality, user interfaces, set of available functionalities, examples and documentation. COMPASS has integrated and harmonized selected features from predecessor projects. Moreover, the quality of the software has been improved in several respects, such as integration of more recent libraries, code re-factoring, and adherence to software quality standards.

An important aspect to be considered, as part of future needs, is the licensing schema. Currently, the licensing of COMPASS is limited to the ESA member states. This is a significant limitation. While, on the one hand, this restriction may help keeping a competitive advantage for ESA member states, on the other hand, it prevents use and cross-fertilization with entities not residing in ESA member states. In the past, the COMPASS Consortium has received requests from entities based outside the ESA member states, including some important US-based industrial companies in the areas of avionics and/or aerospace (e.g., Honeywell USA). These requests could not be granted, due to the licensing limitation. We believe that lifting this limitation could bring important benefits in terms of market penetration and industrial usage. Finally, an open-source versus closed-source (development and release) schema should be evaluated.

## 3.2 Process

Starting with the COMPASS3 project, the development infrastructure of COMPASS has been enhanced, and now includes continuous integration capabilities. Specifically, the code base now resides on a Git repository and can be accessed via a Gitlab repository manager, that enable smooth concurrent development between different teams. Furthermore, automatic testing capabilities have been added, that carry out non-regression testing and benchmarking as part of development. This infrastructure has proved to be very useful for the development of the toolset, and may be further improved in the future. Finally, this schema could be considered as a starting point to address continuous integration of components at HW level.

Concerning the use of COMPASS in system development, currently COMPASS does not provide specific support for process-related activities, such as the development phases foreseen by design standards, e.g. the ECSS standard used in ESA. Areas where the COMPASS toolset could be improved include the possibility to support the generation of artifacts foreseen by the standard and used for design reviews and/or certification. Models produced using COMPASS could also be used as artifacts to support such reviews and certification.

## 3.3 Research

The COMPASS toolset incorporates state-of-the-art technologies for formal modeling and verification. Such technologies are the outcome of a substantial research effort carried out by the research community and by the COMPASS Consortium itself.

The research and technological content of the COMPASS toolset and of the underlying methodologies has been reported in numerous publications (more than 20 publications in international conferences and journals since 2008) and presented in several talks and tutorials. Moreover, its effectiveness has been reported in numerous studies involving real-world and industrial scenarios [18, 42, 35].

Several research challenges remain open. A summary of the most promising directions for future research is discussed in Section 4.3.

## 3.4 Community

The community comprises all the stakeholders that are potentially interested in the usage of the COMPASS toolset, including people from the European Space Agency, national space agencies, industry and academia.

The COMPASS toolset, over the years, has been advertised and demonstrated in several talks and tutorials organized at international conferences and in other venues. Moreover, the COMPASS toolset has been downloaded and evaluated by several entities in academia and industry (about 50 to 100 downloads per year from 2012 onwards).

An additional step towards involvement of the community in the COMPASS effort was the organization of a COMPASS Workshop, of October 23, 2015, at ESA. The objective was to identify hurdles of introducing COMPASS in industrial practice, and discuss and explore ways these hurdles can be taken or circumvented, with potential solutions both in technology as well as process. Representatives of ESA and other agencies, industry and academia participated in the workshop and provided useful feedback.

Despite these past actions and experiences, there is no solid support, yet, for the COMPASS community as a whole, and further actions that go into this direction are needed. Actions may include the organization of workshops and other structured means to get feedback from the community. Currently, we are preparing a questionnaire, to be submitted to all the potential users in the COMPASS community. The questionnaire identifies a list of requirements and/or potential solutions and will help us in prioritizing the actions.

Last but not least, the COMPASS consortium maintains an active relation with the AADL community and, in particular, its standardization committee. This shows up in in various presentations of COMPASS-related activities at committee meetings, which lead to the inclusion of SLIM error modeling concepts such as invisible error states and error state history in the Error Model Annex of ADDL [2].

## 3.5 Integration with ESA Initiatives

In this section we discuss in more detail the relationships between COMPASS and the other ESA initiatives identified in Section 2.5.

The overall picture comprising the ESA initiatives of COMPASS, TASTE, OSRA, and CSSP is shown in Figure 3.1. The COMPASS toolset is intended

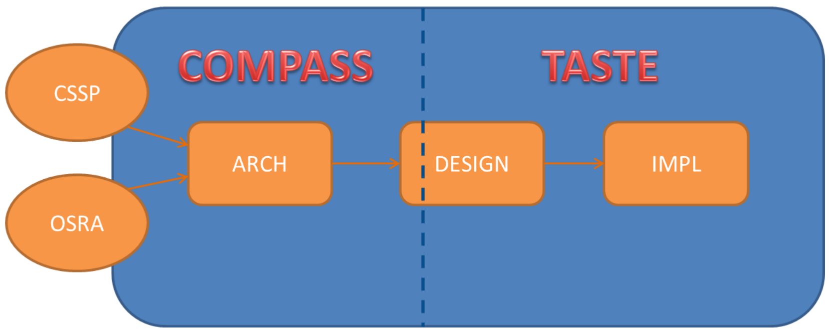

Figure 3.1: COMPASS, TASTE, OSRA and CSSP.

to cover the high level (upstream) requirements and architecture phases of the development process, and to complement the TASTE toolset, which is dedicated to downstream design and implementation phases. The artifacts resulting from the upstream phases should be aligned with the SAVOIR OSRA, and SAVOIR tools could be used to perform useful analyses on the integrated tool chain. Finally, the CSSP can be used as input for the formalization of properties.

The current status with regard to this vision is the following: there is currently no integration between COMPASS and TASTE, although both use AADL for the system view description. However, a preliminary study [22] concerning the model-based design of an energy-system embedded controller has done a preliminary evaluation of TASTE, and identified some requirements of the COMPASS/TASTE integration. No study has been performed to check the alignment of COMPASS with the OSRA, although COMPASS is integrated with OCRA, which is compliant with the Space Component Model, as shown in the FOREVER study. Finally, the CSSP developed in the CATSY project is fully integrated in COMPASS.

# Chapter 4

# The Future of COMPASS

In this chapter, we discuss the most relevant future developments of the COMPASS toolset, research directions and strategies, that can help address the limitations identified in Chapter 3.

## 4.1 Toolset

In this section we discuss the most promising future developments of the COMPASS toolset.

### 4.1.1 SLIM Language

An open direction is to further investigate the streamlining of SLIM with the official AADL language. While COMPASS3 has already improved the alignment between these two languages, there are additional points of investigation. At the core level, the AADL extension mechanism, provided by means of annexes, can continue to be used to implement new SLIM-specific constructs, as to not interfere with the core AADL language. As outlined in Chapter 3, this makes it possible to also work with SLIM models in existing AADL tools.

Other topics include the adaptation of SLIM to the recent version (V2) of the Error Model Annex [2] and the upcoming release (V3) of AADL, which is currently under development with a publication target date of 2018/19. That revision will introduce new concepts such as compositional interfaces, configuration support, virtual memory, and a number of other issues. Of particular interest is a new unified type system, which may consolidate the SLIM type system with that of AADL.

### 4.1.2 Modeling in Other Formalisms

The possibility for end users to model in languages other than SLIM is clearly a plus that could bring significant advantages. In particular, it could enable the use of COMPASS in industrial environments that have a consolidated modeling and design process based on other languages. Moreover, it could facilitate the re-use of existing models.

We intend to approach this issue by creating a front-end in COMPASS for additional input languages. Among them, we may consider languages such as

SMV, Altarica, Simulink, or SysML. Translation back and forth from SLIM and other modeling formalism is technically possible. In order to maintain the alignment and semantic consistency between different models, the underlying COMPASS-internal model or an appropriate meta-model could be used. Moreover, suitable interchange formats will need to be defined.

In general, many input languages are possible, and none will ever be a perfect solution; suitable compromises will have to be sought. In the following section, we sketch an approach to integrate Simulink models.

#### Integration with Simulink

Simulink is a modeling and simulation environment for model-based design widely used in industry. Providing a Simulink frontend for COMPASS will definitely provide ease to the end users for interacting with COMPASS – without the need to model in SLIM. Essentially the Simulink models can be translated to the architectural intermediate representation of COMPASS. The integration of Simulink with COMPASS would also open the opportunity to do contract-based design and verification of the Simulink models. Another benefit of the integration would be the possibility to show results at the Simulink level, e.g. simulating the traces found by the COMPASS verification engines as test-cases in Simulink. However, providing a full Simulink front-end to COMPASS would be challenging – semantics issues and composition rules (synchronous versus asynchronous) should be clarified. As an example, defining a contract in a Simulink model would require to have some custom Simulink blocks – like assume and guarantee blocks. Another challenging example is to deal with the automatic fault extension (a feature of COMPASS), which requires further investigation.

In the opposite direction, the ability to convert SLIM models into Simulink enables the use of Simulink based tooling. A particularly interesting aspect is the ability to create a workflow around Model Driven Engineering (by means of using Simulink for code generation). This direction of transformation also comes with its own challenges, as some of the SLIM semantics do not translate directly into Simulink, e.g. Simulink is more restrictive towards cyclic data dependencies, and does not support continuous behavior like SLIM does.

### 4.1.3 Integration with Design Environments

The integration into design environments, such as Eclipse, has been evaluated. The outcome of the 2015 COMPASS Workshop suggested that the development of the COMPASS toolset would focus on developing and providing services for integration, rather than directly taking on any integration activity. Many interesting design environments and repositories of interest for integration exist, and could therefore be targeted. Examples are the Electronic Data Sheet (EDS), MATLAB, IVY, and AutoFOCUS3.

Another example is the coupling of Capella (architecture only, no behavior) with the modeling capabilities of behavioral aspects in SLIM. In this respect, preliminary work (starting point for evaluation and future developments) has been done by Thales Alenia Space. In particular, it has been investigated during a study aiming at defining a model-based approach for supporting the FDIR process, and some prototyping activities have allowed to couple the COMPASS toolset to Melody Advance (the Thales modeling tool currently released as the open-source software named Capella).

### 4.1.4 Model Validation and Documentation

A functionality that is perceived as very important by the users is the possibility to assess the quality of a formal model. We propose to address this issue in multiple ways. An option is to simulate with respect to an expected scenario. Another possibility is model-to-model comparison, where the model under analysis is compared against a reference model. If the both models are formal, this is similar to sequential equivalence checking. If the reference models is informal (e.g. represented by a set of executions), a possibility is to check if the traces can be re-executed on the model under analysis. Using simulators/emulators is also another option. Finally, deriving structural properties of models such as reachability matrices and presence of deadlocks could also support model validation.

An important issue is change management. In particular, COMPASS should be able to deal with model change and evolution, model re-use, and keep traceability information between different models, and between models and analysis results.

An industrial use case is documentation generation. In this area, we think that it would be possible to generate graphical artifacts from formal models, such as visualizing abstract machines or creating Message Sequence Charts.

### 4.1.5 Scalability

In order to mitigate the issue of scalability, additional investigations could be carried out. This includes the potential incorporation of new engines and algorithms, and tuning of existing routines. This effort is ongoing, for instance a few new algorithms and verification options have been incorporated in COMPASS3, and additional ones may be incorporated in the future. As an example, recent techniques to produce conservative estimates for fault tree probabilities (*anytime FTA*) [17] could be incorporated.

In order to measure and evaluate the progress, it is important to have benchmark sets and case studies of significant size, that can be used to profile the verification engines and find bottlenecks and potential solutions.

### 4.1.6 Case Studies

A strategy for developing comprehensive case studies of industrial size is needed. One possibility that is being pursued is the development of a case study done by the COMPASS Consortium itself, possibly as a result of a joint activity with ESA (e.g., as the outcome of a student internship in ESA). An alternative possibility is to use an openly available case study from another ESA project, such as the satellite case study that was used in the CSSP project or the solar dish case study analyzed in the Contest project [22].

Another direction that is clearly worth investigating is the possibility to directly involve industrial partners in this activity. The development of the case study could be the outcome of an internal evaluation study or the deliverable of a funded project. The feasibility of this schema may depend on interest and commercial strategies of the industrial partner, and available funding.

### 4.1.7 Software and Licenses

We consider COMPASS 3.0 as a consolidated starting point for future developments. After the final release of COMPASS 3.0, our strategy would be to issue periodic releases, when significant new features are incorporated. The documentation, including the user manual and tutorial, will be updated along with the code.

Concerning the licensing limitations of COMPASS, a strategy that has been preliminarily discussed with ESA during the COMPASS3 project could be to open the license also to entities outside the ESA member states, with some restrictions of use, or under some specific terms. For instance, as a counterbalance, licensing to these entities might require granting back the possibility to use some of the outcomes of the evaluation, such as evaluation reports and case studies. The pros and cons of the different schemas (restriction to ESA member states, or open) should be discussed and agreed upon with ESA. Finally, the pros and cons of an open-source versus closed-source (development and release) schema should be evaluated, e.g. the open-source schema could bring benefits in terms of community development.

## 4.2 Process

The development infrastructure, based on continuous integration, used in COMPASS 3.0, will be further extended and improved, with the addition of further non-regression tests and testing capabilities. Moreover, automated testing capabilities for the Graphical User Interface of the COMPASS toolset will be investigated. Continuous integration at HW level can also be considered as an interesting direction for future work.

Regarding the ECSS standard, future extensions include support for the production of artifacts and design documentation that are required by the ECSS standard, as a way to provide inputs for, and effectively improve, preliminary and critical design reviews (PDR, CDR). Along the same line, COMPASS could be extended to support the generation of artifacts that can be used for certification purposes.

Finally, the ECSS standard itself could be updated to take into account advances in model-based development, in particular the 'models-as-deliverables' paradigm. COMPASS could be used to support this view. More in general, the ECSS standard could be updated to take into account the model-based system engineering process, based on formal methods.

## 4.3 Research

There are several research areas that deserve investigation, and that suggest potential extensions of the toolset in terms of additional design or verification capabilities. In this section we have identified a few research challenges and related developments that, in our opinion, are the most beneficial for the final users.

### 4.3.1 Property Validation

Since system properties are the formal counterpart of informal requirements, we do not have a formal specification that can be used to verify their correctness. Therefore, the problem of property validation, i.e., checking if the properties specification captures the requirements intent, is of paramount importance. The problem is exacerbated by the fact that requirements are often ambiguous and flawed, as showed in many studies (e.g., [41]).

COMPASS provides validation of properties based on temporal logic satisfiability (see, e.g., [26]). This enables checking consistency and performing other queries on the property specification. However, it is restricted to linear-time, discrete- or continuous-time, temporal logic. It does not deal with branching-time logics such those involving probabilities. Integrating probabilities in the linear-time settings and solving the satisfiability problem remains an open research problem.

Another challenging problem related to property validation is checking the realizability/implementability of the properties [1], i.e., if the properties can be implemented or constrain somehow the environment so that no implementation can exist. A typical example of unrealizable properties are those that need clairvoyance on the future (e.g., "the output must be true whenever in the future the input will be true"). Providing practical solutions in the case of infinite-state systems is an open research problem.

Finally, an interesting research direction is to exploit synthesis algorithms to enrich debugging information, such as providing stronger/weaker specifications, min/max bounds for delay or probabilities, or more in general synthesizing automatically parts of the property specification. On these lines, the work on contract tightening [23] already uses synthesis to provide tighter versions of the contracts specification and can be further enhanced to be more effective with different kinds of specifications.

### 4.3.2 Contract-Based Fault Injection

In COMPASS, Fault Tree Analysis can be performed either by means of the "traditional" model-based approach, which computes the minimal cut set for a top-level event, or by means of contract-based safety analysis, which produces a hierarchical fault tree that follows the specified contract refinement (and thus the architectural decomposition). The two analyses are currently disconnected: while the model-based safety analysis exploits the error model specification to automatically inject faulty behaviors into the nominal model, the contract-based approach identifies a failure by the fact that the component implementation violates a guarantee or that the component environment fails to satisfy an assumption. Thus, in the contract-based approach, in case of failure, any behavior is possible. This may result in very pessimistic fault trees. An interesting research direction is to find an effective way to inject the faults in the contract specification in order to have degraded assumptions and guarantees in case of failures.

### 4.3.3 Dynamic Fault Tree Analysis

Dynamic fault trees (DFTs) are a well-known extension to standard fault trees that cater for common dependability patterns, such as spare management, functional dependency, and sequencing. Analysis of DFTs relies on extracting an underlying stochastic model, such as Bayesian networks, continuous-time Markov chains (CTMCs), stochastic Petri nets, etc. The expressive power of DFTs is larger than that of static fault trees, and often leads to fault models that are more succinct, and thus better comprehensible. This however, comes at a price: the state-space generation process of DFTs is much more involved. The COMPASS 3.0 toolset allows for the automated generation of DFTs with priority-AND gates. An extension of this fault tree synthesis algorithm to generate full DFTs, with functional dependencies, spare gates, priority-OR gates and so forth would be an interesting direction. This can be complemented with techniques to generate state spaces for DFTs in a comprehensive manner using various reduction techniques from the field of model checking – such as symmetry reduction, partial-order reduction – and static analysis techniques on DFTs. Initial experiments with these state-space generation techniques give very promising results [50]. Partial state-space generation could give under- and over-approximations of measures-of-interest on DFTs, and could boost the generation process further.

A related study that may be taken into account is the ESA TRP called VeriFIM [49] (Verification of Failure Impact by Model Checking), that addressed the on-board prognosis perspective (pro-active FDIR). It was based on the system DFT modeling and the translation to the Bayesian Networks, which were then, off-line, translated to the Junction Trees for On-Board Prognostic FDIR computations.

### 4.3.4 FDIR Design

The area of diagnosability and fault detection and identification (FDI) design is particularly challenging. Recent work [15, 16] has addressed the extension of diagnosis and FDI to incorporate the notion of delay, and to address cases where diagnosability cannot always be guaranteed for all system executions, but only locally. In [16], related aspects of an FDI design can be specified using a general framework and a language for specification patterns, based on temporal epistemic logic. Verification can be performed using an epistemic model checker or reduced to standard temporal logic model checking based on the twin-plant approach [25]. Incorporation of these features in COMPASS, and extensions to continuous models, is still an open point. Moreover, the extension of this framework to fault recovery and recoverability is still missing.

Another interesting research area concerns the analysis of observability requirements for diagnosis, and the synthesis of a set of observables that are sufficient to ensure diagnosability [13]. It is possible to rank configurations of observables based on cost, minimality, and diagnosability delay, thus helping designers in finding the most appropriate configuration. Moreover, the automatic synthesis of FDIR components has been considered in two COMPASS-related projects, namely AUTOGEF [5] and FAME [10], but is not yet available in COMPASS. A full-fledged methodology, process, and tool support for the FDIR design process, starting from the FDIR requirements and exploiting the results of safety assessment, would be extremely beneficial. Preliminary results from AUTOGEF and FAME are encouraging, however a substantial effort is needed to consolidate these results and make them more mature.

Another error modeling concept to be investigated further are Timed Failure Propagation Graphs (TFPG), which enable a precise description of how and when failures originating in one part of a system affect other parts – a fundamental feature for successfully designing contingency mechanisms. A TFPG is a graph-like model that accounts for the temporal progression of failures and for the causality between failure effects, taking into consideration time delays and system (re-)configuration. The nodes of a TFPG represent either failures or discrepancies (representing anomalous behaviors), whereas edges represent propagation links, labeled with timing information (minimum and maximum propagation time) and modes (system modes enabling the propagation) – see an example in Fig. 4.1. New COMPASS functionalities for TFPG analysis and synthesis, such as the ones described in [14, 12, 9], would be beneficial. In particular, automatic *tightening* of TFPG nodes, coupled with the synthesis of the TFPG graph, may be used to automatically produce a complete and tight TFPG from a system model, given the definition of the TFPG nodes.

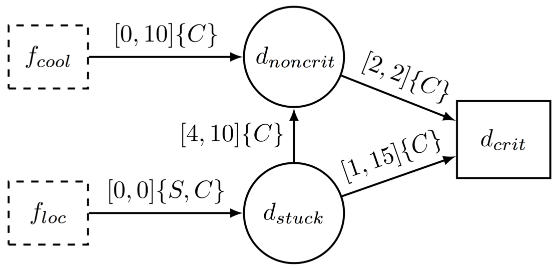

Figure 4.1: An example TFPG

### 4.3.5 Design Space Exploration

An important problem, which is especially relevant in the early phases of the design of safety-critical systems, is the ability to analyze the safety of alternative design solutions, comparing how different functional allocations impact the overall reliability of the system. Different solutions may be evaluated using formal techniques ranging from model checking to FTA.

Previous work in this area has been done in collaboration with NASA in the context of the next generation air traffic control system for the United States [42, 35]. Aspects that have been evaluated are the allocation of separation assurance capabilities (traditional ground-separated aircraft, and self-separated aircraft) and the required communication between agents. To deal with the extremely high number of designs (more than 20,000), a new compositional, modular, and parameterized approach was used. Such approach combined model checking with contract-based design to automatically generate large numbers of models from a possible set of components and their implementations. Results were validated by NASA system designers, and helped identify novel as well as known problematic configurations. We plan to integrate and evaluate similar techniques in COMPASS, to enable the exploration of different design solutions for aerospace systems.

### 4.3.6 Parametric Models

COMPASS uses both qualitative and probabilistic model checking as underlying techniques to perform different kinds of analyses such as the verification of the model's correctness or the evaluation of its performability characteristics where, given an AADL model with associated error probabilities, the likelihood of a system failure occurring within a given deadline is determined. In many cases, some values such as the execution time of a thread or the probabilities of basic faults are not known, or at best can be estimated by lower and upper bounds. It would therefore be worthwhile to consider *parametric* models, in which such values are left open, symbolically represented by some variables, called *parameters*.

*Parameter synthesis* focuses on automatically computing the region of parameter values such that the resulting model satisfies its correctness or performability requirements, see Figure 4.2. Although this problem is inherently harder than model checking, in some cases model-checking techniques can be generalized to solve the synthesis problem (see for example [24, 11]). Also in the case of probabilistic model checking, first results indicate that for a limited number of parameters, solutions are feasible and scalable [30, 39]. They would allow to derive quality requirements for electronic [46] or mechanical parts [43] or software components [6] of a system to be developed.

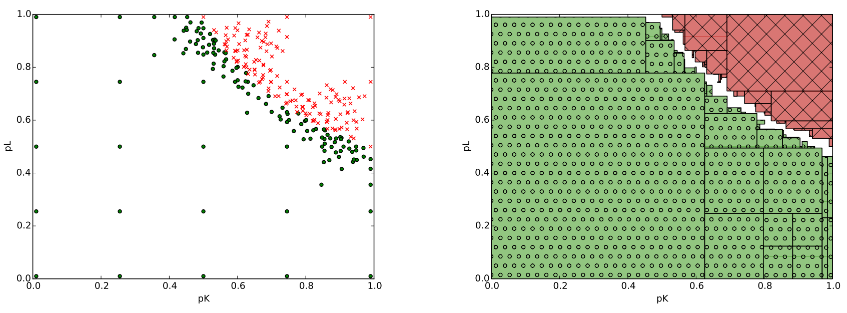

Figure 4.2: Possible result for analysis of parametric model with two parameters, with green regions indicating where a safety threshold has been met.

A related problem is *model repair*, where one tries to tune the parameters of a given model such that the resulting model satisfies a given correctness or performability requirements. Regarding probability parameters, current approaches only consider changes of the transition probabilities, whereas modifications of the underlying topological structure are not considered. Different methods exist, such as global repair [8] and iterative local repair [45]. The repair of AADL models can potentially lead to the synthesis of models that are compliant with the given correctness or performability requirements. Similarly, the techniques described in [23], which use parameter synthesis to find tighter contract refinements, can be potentially generalized to synthesize correct contract refinements.

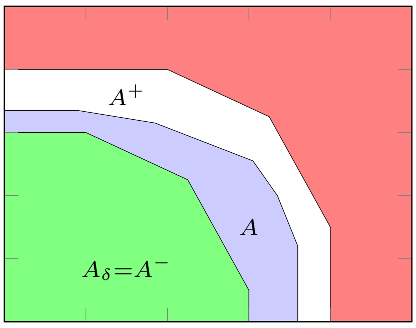

Figure 4.3: Possible result for multi-objective verification, indicating the range of possible values that achieve the goal. From the bottom-left the areas indicate an under-approximation $A^-$, the actual range $A$ and an over-approximation $A^+$. Values outside $A^+$ can never satisfy the objectives.

### 4.3.7 Multi-Objective Verification

Besides the correctness of their functional behavior, systems are required to perform well. *Performance* of a system can be measured by, e.g., its average and peak energy consumption, construction costs, and its availability and reliability. These measures are often contradictory: using more power for data transmission typically increases the reliability of communication, but also induces a higher energy consumption. Less obvious mutual dependencies can emerge: optimizing a system for (long-run) availability might reduce the (short-term) reliability.

The problem of finding optimal solutions with respect to different criteria has been explored in [11], where an approach based on the IC3 routine has been presented, and demonstrated on applications from diagnosability synthesis and product-line engineering.

In the context of probabilistic analysis, *multi-objective model checking* can be employed. This is a fully automatic technique by which, based on a model of the system under consideration and some measures-of-interest, a so-called Pareto curve is deduced [34]. The latter gives an (often graphical, see Figure 4.3) representation of the optimal strategy for resolving non-deterministic choices in the system with respect to a given weighting of these measures. Currently, only Markov Decision Processes can be handled by this technique [32, 40]. While these support non-deterministic choices (to be optimised) and discrete probabilities, they lack continuously distributed random delays, which are typically used to describe, e.g., mechanical wear or other sources of failures.

*Markov Automata* [36] constitute a highly expressive formalism which extend Markov Decision Processes by such random delays. They are used within the performability engine of COMPASS 3.0, which is able to cope with optimizing single measures on Markov Automata [36] such as minimizing the probability to enter an error state within a given deadline. An extension of multi-objective model checking to Markov automata would enable a trade-off analysis of several measures-of-interest of AADL models. For example, the trade-off between using more unreliable system components and the expected time until a system failure might be analyzed.

### 4.3.8 Model-Based Testing

All analysis possibilities supported by the COMPASS 3.0 toolset are model based. That is, all analysis results are derived from the AADL model under consideration. No guarantees are provided for given or possible realizations of these models. Thus, there are currently no (or very limited) means to check the conformance of a hardware/software implementation with respect to the AADL model.

Model-based testing [20] is a technique that in principle allows for doing this. The underlying idea of this approach is to steer the automated test generation process by the AADL model. The tests are then turned into executable test cases for a given implementation (commonly referred to as the system under test); such test cases thus are depending on the implementation at hand. As the model is used to steer the test generation, testing is sound – if an executable test fails, the implementation does not conform to the AADL model. In theory it is also complete – if an implementation passes all tests, the implementation is compliant. In practice, however, not all tests can be run, and automated test selection becomes relevant. Finally, an advantage of model-based testing is that changes in the design/model can be reflected directly and automatically in the generated tests.

The challenge is to exploit model-based testing for AADL models, tailor it to the features of AADL (such as component-wise testing), and extend the existing model-based testing approaches and theories with notions such as real-time, hybrid behavior, and randomness. This could give rise to a fully automated technique for checking the compliance of HW/SW implementations against AADL models. This functionality could be also integrated in the TASTE environment.

Finally, model-based testing, and connection with implementation-level deployment, could be coupled with the following techniques: property-preserving model transformations and – following the contract-based approach – the generation of "test obligations" on the implementation that enforce contracts to hold.

## 4.4 Community

We have identified the following measures to increase the involvement of the community in the COMPASS development, and improve publicity and advertisement.

- An outcome of the COMPASS3 project has been the setup of a comprehensive web site [29], with material on the COMPASS initiative and toolset, past projects, examples and case studies, publications, etc. The web site will be further updated and improved.
- A COMPASS-related workshop, namely the *Workshop on Model Based System and Software Engineering (MBSSE) – Future Directions*, has been organized on December 8th, 2016 at ESA. As for the COMPASS Workshop in 2015, different stakeholders have been invited. At the workshop, we have advertised and demonstrated the new COMPASS 3.0 toolset, we have collected users' requirements and needs, and received feedback on the roadmap, that has been incorporated into this document. Further

feedback from the community will be sought and will help in further tuning the roadmap and steering future developments.

- In September 2017, FBK will host the SEFM [48], IMBSA [38] and Safecomp [47] conferences. These conferences may offer good opportunities to disseminate the work done in the COMPASS initiative and collect feedback. Possibly, a cross-conference workshop could be organized.
- In order to get further feedback from the community, we will finalize the questionnaire (compare Section 3.4), to be submitted to all the potential users. The outcome of the questionnaire will help us in prioritizing the needs and potential solutions.
- The COMPASS3 project has produced training material in the form of a hands-on tutorial. Further training material, such as slides and material for courses, could be prepared. Also, the COMPASS tutorial could further elaborate on modeling guidelines and best practices.
- Further advertisement actions, such as creating a Wikipedia page for COMPASS, or an entry in the ResearchGate portal, could be targeted.
- The collaboration with the AADL standardization committee will be continued, both triggering extensions of AADL by concepts that are relevant for aerospace applications and adapting SLIM to changes in AADL and its supporting annexes. In particular, it is planned to initiate a joint research project to investigate enhanced mechanisms for integrating the nominal and error behavior of a system.

## 4.5 Integration with ESA Initiatives

In order to pursue the integration of COMPASS with the other ESA initiatives identified in the previous chapters, we foresee the following directions.

- Tight integration of COMPASS and TASTE so that the AADL model analyzed with COMPASS is further detailed in TASTE for the deployment. This would enable merging the functionalities of requirements analysis, system design and system implementation into a unique tool chain. To achieve this, we need to adopt a compatible description of the component behavior and adopt a common semantics for the components interaction. Based on this common semantics, the integration with TASTE could also exploit the code generation capabilities of TASTE.
- Ensure the compliance of the AADL models used in COMPASS and TASTE with the Space Component Model of OSRA. Also, take into account the detailed data handling architecture.
- Enhance COMPASS with the library of components used in the OSRA.
- Enhance the OSRA components with CSSP properties.
- Investigate the possibility to have an FDIR reference architecture, along the sames lines as the OSRA initiative, and supported by the COMPASS toolset.

# Chapter 5

# Planning

Based on the objectives, needs and potential future activities identified in, respectively, Chapter 2, Chapter 3 and Chapter 4, in this chapter we identify a set of actions to address the needs and reach the objectives, and we discuss their priorities.

## 5.1 Actions

We split the set of actions into "block-removal" actions, i.e., short-term actions that aim at mitigating/removing limitations/blocking aspects/bottlenecks that currently affect the use of COMPASS, and "new-feature" actions, i.e., longer-term actions that genuinely aim at extending the toolset with new features and functionalities.

### 5.1.1 Block-Removal Actions

**A1** Identify opportunities to make SLIM further compliant with AADL.

**A2** Make SLIM compliant with AADL Error Model Annex V2, investigate enhanced mechanisms for integrating the nominal and error behavior.

**A3** Develop a case study of significant size.

**A4** Identify opportunities for industrial evaluation and development of case studies in industry.

**A5** Develop a set of benchmarks.

**A6** Identify a new licensing schema, open to non-ESA member states.

**A7** Extend the COMPASS continuous integration environment.

**A8** Extend automated testing to GUI features.

**A9** Dissemination at ESA Workshop in December 2016: collect further feedback on COMPASS roadmap.

**A10** Add further material to the COMPASS web site.

**A11** Create further training material: slides, courses, PhD schools.

**A12** Create Wikipedia page for COMPASS.

**A13** Create entry in ResearchGate for COMPASS.

**A14** Finalize questionnaire for end users, collect and evaluate results.

### 5.1.2 New-Feature Actions

**A15** Make SLIM compliant with AADL V3.

**A16** Investigate use of OSATE for SLIM models.

**A17** Develop frontend for Altarica.

**A18** Develop frontend for Simulink.

**A19** Develop frontend for SysML.

**A20** Integrate COMPASS with the Eclipse design environment.

**A21** Integrate COMPASS with the Capella design environment.

**A22** Implement converter from SLIM models into Simulink.

**A23** New functionality: model simulation with respect to scenarios.

**A24** New functionality: model-to-model comparison.

**A25** New functionality: displayer for finite state machines.

**A26** New functionality: anytime FTA.

**A27** Generate ECSS-compliant artifacts and documentation.

**A28** Generate certification-oriented artifacts.

**A29** New feature: validation of probabilistic properties.

**A30** New feature: property realizability.

**A31** New feature: property synthesis.

**A32** Integrate flat FTA and contract-based (hierarchical) FTA.

**A33** New functionality: extended DFT analysis.

**A34** New functionality: FDI extended specification of diagnosability conditions and local diagnosability.

**A35** New functionality: extend FDI framework to recoverability.

**A36** New functionality: synthesis of observables.

**A37** New functionality: synthesis of FDI models.

**A38** Review FDIR design process and use of TFPG analyses.

**A39** New functionality: TFPG tightening/synthesis.

**A40** New functionality: design space exploration.

**A41** New functionality: parametric error models, model repair/synthesis of parameters for performability analysis.

**A42** New functionality: multi-objective verification.

**A43** New functionality: model-based testing.

**A44** Dissemination at SEFM/IMBSA/Safecomp 2017.

**A45** Dissemination to the AADL standardization committee.

**A46** Publish in international journals and conferences.

**A47** Integrate COMPASS and TASTE: support for multiple behavioral models, reliability models, deployment and code generation.

**A48** New functionality: software model checking for TASTE.

**A49** Make COMPASS models compliant with OSRA.

**A50** ADD OSRA library of components to COMPASS.

**A51** ADD CSSP properties to OSRA components.

**A52** Define FDIR reference architecture.

**A53** New functionality: change management, model change and evolution.

## 5.2 Traceability of Actions to Objectives

In this section we trace the actions identified in Section 5.1 to the objectives they aim to reach. The traceability information is shown in Table 5.1.

| | Objectives | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Tool | | | Process | | | Research | | | Community | | | Integration | | |
| | T1: Usability | T2: Higher TRL | T3: Accessibility | P1: Infrastructure | P2: ECSS Compliance | P3: Certification | R1: Applicability | R2: Scalability | R3: Dissemination | C1: Visibility | C2: Market penetration | C3: Industrial usage | I1: TASTE | I2: OSRA | I3: FDIR |
| A1: AADL compl. | X | | | | | | | | | X | X | X | | | |
| A2: EM Annex V2 | X | | | | | | | | | X | X | X | | | |
| A3: Case study | | X | | | | | X | X | X | X | X | X | | | |
| A4: Ind. Evaluation | | X | | | | X | X | X | X | X | X | X | | | |
| A5: Benchmarks | | X | | | | | X | X | | | | | | | |
| A6: New license | | | X | | | | | | X | X | X | X | | | |

| | Objectives | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Tool | | | Process | | | Research | | | Community | | | Integration | | |
| | T1: Usability | T2: Higher TRL | T3: Accessibility | P1: Infrastructure | P2: ECSS Compliance | P3: Certification | R1: Applicability | R2: Scalability | R3: Dissemination | C1: Visibility | C2: Market penetration | C3: Industrial usage | I1: TASTE | I2: OSRA | I3: FDIR |
| A7: Extend CI | | X | | X | | | | | | | | | | | |
| A8: GUI testing | | X | | X | | | | | | | | | | | |
| A9: ESA Workshop | | | | | | | | | X | X | X | X | | | |
| A10: Web site | | | X | | | | | | X | X | X | X | | | |
| A11: Training mater. | X | | | | | | | | X | X | X | X | | | |
| A12: Wikipedia | | | X | | | | | | X | X | X | X | | | |
| A13: ResearchGate | | | X | | | | | | X | X | X | X | | | |
| A14: Questionnaire | X | X | | | | | | | X | X | X | X | | | |
| A15: AADL V3 | X | | | | | | | | | X | X | X | | | |
| A16: OSATE | X | | | | | | X | | | | X | X | | | |
| A17: Altarica front. | X | X | | | | | X | | X | X | X | X | | | |
| A18: Simulink front. | X | X | | | | | X | | X | X | X | X | | | |
| A19: SysML front. | X | X | | | | | X | | X | X | X | X | | | |
| A20: Eclipse integr. | X | X | | | | | X | | X | X | X | X | | | |
| A21: Capella integr. | X | X | | | | | X | | X | X | X | X | | | |
| A22: Simulink conv. | X | X | | | | | X | | X | X | X | X | | | |
| A23: Scenario simul. | X | X | | | | | X | | | | X | X | | | |
| A24: Model compar. | X | X | | | | | X | | | | X | X | | | |
| A25: FSM displayer | X | X | | | | | X | | | | X | X | | | |
| A26: Anytime FTA | X | X | | | | | X | X | | | X | X | | | |
| A27: ECSS document. | X | X | | | X | X | X | | X | X | X | X | | | |
| A28: Certif. artif. | X | X | | | X | X | X | | X | X | X | X | | | |
| A29: Prob. prop. v. | X | X | | | | | X | | | | X | X | | | |
| A30: Realizability | X | X | | | | | X | | | | X | X | | | |
| A31: Prop. synth. | X | X | | | | | X | | | | X | X | | | |
| A32: Flat/hier. FTA | X | X | | | | | X | X | | | X | X | | | |
| A33: Extended DFT | X | X | | | | | X | | | | X | X | | | |
| A34: Extended FDI | X | X | | | | | X | | | | X | X | | | |
| A35: Recoverability | X | X | | | | | X | | | | X | X | | | |
| A36: Observ. synth. | X | X | | | | | X | | | | X | X | | | |
| A37: FDI synth. | X | X | | | | | X | | | | X | X | | | |
| A38: FDIR process | X | X | | | | | X | | | | X | X | | | X |
| A39: TFPG tighten. | X | X | | | | | X | | | | X | X | | | |
| A40: Design S. Expl. | X | X | | | | | X | | | | X | X | | | |
| A41: Param. error m. | X | X | | | | | X | | | | X | X | | | |
| A42: Multi-obj. ver. | X | X | | | | | X | | | | X | X | | | |
| A43: Model-b. testing | X | X | | | | | X | | | | X | X | | | |
| A44: Confer. 2017 | | | | | | | | | X | X | X | X | | | |

|  | Objectives |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | Tool |  |  | Process |  |  | Research |  |  | Community |  |  | Integration |  |  |
|  | T1: Usability | T2: Higher TRL | T3: Accessibility | P1: Infrastructure | P2: ECSS Compliance | P3: Certification | R1: Applicability | R2: Scalability | R3: Dissemination | C1: Visibility | C2: Market penetration | C3: Industrial usage | I1: TASTE | I2: OSRA | I3: FDIR |
| A45: AADL committ. |  |  |  |  |  |  |  |  | X | X | X | X |  |  |  |
| A46: Publications |  |  |  |  |  |  |  |  | X | X | X | X |  |  |  |
| A47: TASTE integr. | X | X |  |  | X | X | X |  | X | X | X | X | X |  |  |
| A48: Software MC | X | X |  |  |  |  | X |  |  |  | X | X | X |  |  |
| A49: OSRA compl. | X | X |  |  | X | X | X | X | X | X | X | X |  | X | X |
| A50: OSRA library | X | X |  |  | X | X | X | X | X | X | X | X |  | X | X |
| A51: CSSP in OSRA | X | X |  |  |  |  | X |  |  |  | X | X |  | X |  |
| A52: FDIR ref. arch. | X | X |  |  | X | X | X |  | X | X | X | X |  |  | X |
| A53: Change manag. | X | X |  |  | X | X | X |  |  |  | X | X |  |  | X |

Table 5.1: Traceability of actions to objectives.

## 5.3 Action Priorities

In this section we rank actions, based on priorities. This is done in Table 5.2.

|  | Priority | Notes |
|---|---|---|
| A1: AADL compl. | medium | Needs further analysis |
| A2: EM Annex V2 | medium | Needs further analysis |
| A3: Case study | high | Blocking for improving scalability |
| A4: Ind. Evaluation | high | May need funding schema |
| A5: Benchmarks | high | Blocking for improving scalability |
| A6: New license | high | Blocking for use in non-ESA member states |
| A7: Extend CI | medium | Many features already available |
| A8: GUI testing | low/medium | Currently done via manual TPs |
| A9: ESA Workshop | high | - |
| A10: Web site | medium | Already contains a lot of material |
| A11: Training mater. | medium/high | Tutorial can be a starting point |
| A12: Wikipedia | medium | - |
| A13: ResearchGate | medium | - |
| A14: Questionnaire | high | - |
| A15: AADL V3 | medium | AADL V3 to be delivered in 2018/2019 |
| A16: OSATE | medium | - |
| A17: Altarica front. | high | - |
| A18: Simulink front. | high | - |
| A19: SysML front. | high | - |
| A20: Eclipse integr. | medium/high | - |

| | Priority | Notes |
|---|---|---|
| A21: Capella integr. | medium/high | - |
| A22: Simulink conv. | medium | - |
| A23: Scenario simul. | medium | - |
| A24: Model compar. | medium | - |
| A25: FSM displayer | medium | Prototype already available |
| A26: Anytime FTA | medium/high | Already available, to be integrated |
| A27: ECSS document. | medium/high | - |
| A28: Certif. artif. | low/medium | Important, but future work |
| A29: Prob. prop. v. | medium | - |
| A30: Realizability | medium | - |
| A31: Prop. synth. | medium | - |
| A32: Flat/hier. FTA | medium/high | - |
| A33: Extended DFT | medium | - |
| A34: Extended FDI | medium | Already available, to be integrated |
| A35: Recoverability | medium/high | - |
| A36: Observ. synth. | medium | Already available, to be integrated |
| A37: FDI synth. | low-medium | Prototype available, but not fully mature |
| A38: FDIR process | high | Preliminary work done in FAME |
| A39: TFPG tighten. | medium | Already available, to be integrated |
| A40: Design S. Expl. | medium/high | Previous work done with NASA |
| A41: Param. error m. | medium | - |
| A42: Multi-obj. ver. | medium | - |
| A43: Model-b. testing | medium | - |
| A44: Confer. 2017 | high | Talks, demos, tutorial, joint workshop |
| A45: AADL committ. | medium/high | - |
| A46: Publications | medium | - |
| A47: TASTE integr. | high | - |
| A48: Software MC | medium | - |
| A49: OSRA compl. | medium/high | - |
| A50: OSRA library | medium/high | - |
| A51: CSSP in OSRA | medium/high | - |
| A52: FDIR ref. arch. | medium | - |
| A53 Change manag. | high | - |

Table 5.2: Action priority.

# Bibliography

[1] A., Rosner, R.: On the Synthesis of a Reactive Module. In: Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989. pp. 179–190 (1989)

[2] Architecture Analysis & Design Language (AADL) Annex, Volume 1, Annex E: Error Model Annex. SAE Standard AS5506/1A, International Society of Automotive Engineers (Sep 2015)

[3] Architecture Analysis & Design Language (AADL). SAE Standard AS5506, International Society of Automotive Engineers (Nov 2004)

[4] Architecture Analysis & Design Language (AADL) (rev. B). SAE Standard AS5506B, International Society of Automotive Engineers (Sep 2012)

[5] Alaña, E., Naranjo, H., Yushtein, Y., Bozzano, M., Cimatti, A., Gario, M., de Ferluc, R., Garcia, G.: Automated generation of FDIR for the COMPASS integrated toolset (AUTOGEF). In: Proc. DASIA 2012. vol. ESA SP 701 (2012)

[6] Alonso, J., Grottke, M., Nikora, A.P., Trivedi, K.S.: An empirical investigation of fault repairs and mitigations in space mission system software. In: Proc. DSN 2013. pp. 1–8. IEEE (2013)

[7] The AUTOGEF project, `http://autogef-project.fbk.eu`

[8] Bartocci, E., Grosu, R., Katsaros, P., Ramakrishnan, C.R., Smolka, S.A.: Model repair for probabilistic systems. In: Proc. TACAS 2011. LNCS, vol. 6605, pp. 326–340. Springer (2011)

[9] Bittner, B., Bozzano, M., Cimatti, A.: Automated synthesis of timed failure propagation graphs. In: Proc. IJCAI 2016. pp. 972–978. AAAI Press (2016)

[10] Bittner, B., Bozzano, M., Cimatti, A., de Ferluc, R., Gario, M., Guiotto, A., Yushtein, Y.: An integrated process for FDIR design in aerospace. In: Proc. IMBSA 2014. LNCS, vol. 8822, pp. 82–95. Springer (2014)

[11] Bittner, B., Bozzano, M., Cimatti, A., Gario, M., Griggio, A.: Towards pareto-optimal parameter synthesis for monotonic cost functions. In: Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design. pp. 9:23–9:30. FMCAD '14, FMCAD Inc, Austin, TX (2014), `http://dl.acm.org/citation.cfm?id=2682923.2682935`

[12] Bittner, B., Bozzano, M., Cimatti, A., Zampedri, G.: Automated verification and tightening of failure propagation models. In: Proc. of AAAI 2016. pp. 3724–3730 (2016)

[13] Bittner, B., Bozzano, M., Cimatti, A., Olive, X.: Symbolic synthesis of observability requirements for diagnosability. In: Proc. AAAI-12 (2012)

[14] Bozzano, M., Cimatti, A., Gario, M., Micheli, A.: SMT-based validation of timed failure propagation graphs. In: Proc. AAAI 2015. pp. 3724–3730 (2015)

[15] Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal design of fault detection and identification components using temporal epistemic logic. In: Proc. TACAS 2014. LNCS, vol. 8413, pp. 46–61. Springer (2014)

[16] Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal design of asynchronous fault detection and identification components using temporal epistemic logic. Logical Methods in Computer Science 11(4), 1–33 (2015)

[17] Bozzano, M., Cimatti, A., Mattarei, C., Griggio, A.: Efficient anytime techniques for model-based safety analysis. In: Proc. CAV 2015. LNCS, vol. 9206, pp. 603–621. Springer (2015)

[18] Bozzano, M., Cimatti, A., Pires, A.F., Jones, D., Kimberly, G., Petri, T., Robinson, R., Tonetta, S.: Formal Design and Safety Analysis of AIR6110 Wheel Brake System. In: CAV. pp. 518–535 (2015)

[19] Bozzano, M., Cimatti, A., Katoen, J.P., Katsaros, P., Mokos, K., Nguyen, V.Y., Noll, T., Postma, B., Roveri, M.: Spacecraft early design validation using formal methods. Reliability Engineering and System Safety 132, 20–35 (2014)

[20] Broy, M., Jonsson, B., Katoen, J.P., Leucker, M., Pretschner, A. (eds.): Model-Based Testing of Reactive Systems: Advanced Lectures. No. 3472 in LNCS, Springer (2005)

[21] The CATSY project, `https://es.fbk.eu/projects/catsy`

[22] Cavada, R., Cimatti, A., Crema, L., Roccabruna, M., Tonetta, S.: Model-Based Design of an Energy-System Embedded Controller using Taste. In: Proc. of FM'2016 (2016), to be published

[23] Cimatti, A., Demasi, R., Tonetta, S.: Tightening a contract refinement. In: Proc. SEFM 2016. pp. 386–402 (2016)

[24] Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: Parameter synthesis with IC3. In: FMCAD. pp. 165–168 (2013)

[25] Cimatti, A., Pecheur, C., Cavada, R.: Formal verification of diagnosability via symbolic model checking. In: Proc. IJCAI 2003. pp. 363–369. Morgan Kaufmann (2003)

[26] Cimatti, A., Roveri, M., Susi, A., Tonetta, S.: Validation of requirements for hybrid systems: A formal approach. ACM Trans. Softw. Eng. Methodol. 21(4), 22 (2012)

[27] Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: Infinite-state invariant checking with IC3 and predicate abstraction. Formal Methods in System Design pp. 1–29 (2016), `http://dx.doi.org/10.1007/s10703-016-0257-4`

[28] The CITADEL project, `http://www.citadel-project.org`

[29] The COMPASS project, `http://www.compass-toolset.org`

[30] Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.P., Abraham, E.: PROPhESY: A probabilistic parameter synthesis tool. In: Proc. CAV 2015. LNCS, vol. 9206, pp. 214–231. Springer (2015)

[31] The D-MILS project, `http://www.d-mils.org`

[32] Etessami, K., Kwiatkowska, M.Z., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Logical Methods in Computer Science 4(4) (2008)

[33] The FAME project, `http://fame-project.fbk.eu`

[34] Forejt, V., Kwiatkowska, M., Parker, D.: Pareto curves for probabilistic model checking. In: Proc. ATVA 2012. LNCS, vol. 7561, pp. 317–332. Springer (2012)

[35] Gario, M., Cimatti, A., Mattarei, C., Tonetta, S., Rozier, K.: Model Checking at Scale: Automated Air Traffic Control Design Space Exploration. In: Proc. CAV 2016. LNCS, vol. 9780, pp. 3–22. Springer (2016)

[36] Guck, D., Hatefi, H., Hermanns, H., Katoen, J.P., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: Proc. QEST 2013. LNCS, vol. 8054, pp. 55–71. Springer (2013)

[37] The HASDEL project, `http://hasdel-project.fbk.eu`

[38] IMBSA 2017: International Symposium on Model-Based Safety and Assessment, `http://imbsa2017.fbk.eu`

[39] Jansen, N., Corzilius, F., Volk, M., Wimmer, R., Abraham, E., Katoen, J.P., Becker, B.: Accelerating parametric probabilistic verification. In: Proc. QEST 2014. LNCS, vol. 8657, pp. 404–420. Springer (2014)

[40] Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Compositional probabilistic verification through multi-objective model checking. Information and Computation 232, 38–65 (2013)

[41] Lutz, R.: Analyzing software requirements errors in safety-critical, embedded systems. In: Proceedings of IEEE International Symposium on Requirements Engineering, RE 1993, San Diego, California, USA, January 4-6, 1993. pp. 126–133 (1993)

[42] Mattarei, C., Cimatti, A., Gario, M., Tonetta, S., Rozier, K.: Comparing Different Functional Allocations in Automated Air Traffic Control Design. In: Proc. FMCAD 2015. pp. 112–119. FMCAD Inc (2015)

[43] Nonelectronic parts reliability data (NPRD-2016). Tech. rep., Quanterion Solutions Inc. (2015), https://www.quanterion.com/product/publications/nonelectronic-parts-reliability-data-publication-nprd-2016/

[44] OSATE – the Open Source AADL Tool Environment, www.aadl.info/tool/osate.html

[45] Pathak, S., Abraham, E., Jansen, N., Tacchella, A., Katoen, J.P.: A greedy approach for the efficient repair of stochastic models. In: Proc. NFM 2015. LNCS, vol. 9058, pp. 295–309. Springer (2015)

[46] Reliability Prediction of Electronic Equipment. No. MIL-HDBK-217F in Military standardization handbook, Department of Defense, USA (1995), http://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=53939

[47] SAFECOMP 2017: International Conference on Computer Safety, Reliability and Security, http://safecomp17.fbk.eu

[48] SEFM 2017: International Conference on Software Engineering and Formal Methods, http://sefm17.fbk.eu

[49] The VeriFIM study, http://people.unipmn.it/dcr/verifim

[50] Volk, M., Junges, S., Katoen, J.P.: Advancing dynamic fault tree analysis – get succinct state spaces fast and synthesise failure rates. In: Proc. SAFECOMP 2016. LNCS, vol. 9922, pp. 253–265. Springer (2016)