The xSAP Safety Analysis Platform

B. Bittner, M. Bozzano, R. Cavada, A. Cimatti,
M. Gario, A. Griggio, C. Mattarei, A. Micheli, and G. Zampedri

Fondazione Bruno Kessler, Trento, Italy

Abstract. This paper describes the xSAP safety analysis platform. xSAP provides several model-based safety analysis features for finite- and infinite-state synchronous transition systems. In particular, it supports library-based definition of fault modes, an automatic model extension facility, generation of safety analysis artifacts such as Dynamic Fault Trees (DFTs) and Failure Mode and Effects Analysis (FMEA) tables. Moreover, it supports probabilistic evaluation of Fault Trees, failure propagation analysis using Timed Failure Propagation Graphs (TFPGs), and Common Cause Analysis (CCA). xSAP has been used in several industrial projects as verification back-end, and is currently being evaluated in a joint R&D Project involving FBK and The Boeing Company.

1 Introduction

In recent years, there has been a growing industrial interest in model-based safety assessment techniques (MBSA) [1,2,3,4,5] and their application. These methods are based on a single safety model of a system, and analyses are carried out with a high degree of automation, thus reducing the most tedious and error-prone activities that today are carried out manually. Formal verification tools based on model checking have been extended to automate the generation of artifacts such as Fault Trees and FMEA tables, which are required for certification of safety critical systems – see, e.g.,[6,7,8].

xSAP is a platform for Model-Based Safety Analysis (MBSA), which provides a variety of features. First, it enables the definition of fault modes, based on a customizable fault library. Second, it implements automatic model extension, namely the possibility to automatically extend a system model with the fault definitions retrieved from the library. Third, it implements a full range of safety analyses, including Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), failure propagation analysis using Timed Failure Propagation Graphs (TFPGs), and Common Cause Analysis (CCA). Finally, xSAP implements a family of effective routines for such analyses, based on state-of-the-art model checking techniques, including BDD-, SAT- and SMT-based techniques.

xSAP is currently the core verification engine for many other tools, including industrial ones. It has been used in several industrial projects, including COMPASS [9], AUTOGEF [10], FAME [11] and HASDEL [12], funded by the European Space Agency. Moreover, xSAP is currently being used in a joint research and development project between FBK and The Boeing Company [13].

xSAP is being developed by FBK, and it is currently distributed with a free license for academic research purposes and non-commercial applications. xSAP can be downloaded from `http://xsap.fbk.eu`.

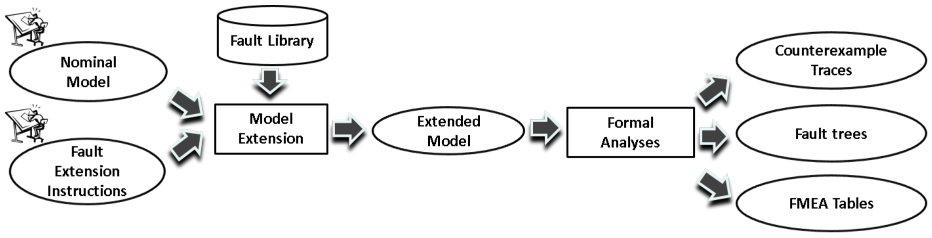


Fig. 1. The XSAP main flow.

Related Work. The system closest to XSAP is the FSAP platform [14], which is no longer maintained. XSAP extends FSAP along several directions. First, FSAP was limited to finite-state systems, while XSAP can handle infinite-state ones. Second, XSAP provides more general and customizable libraries to define fault modes and their dynamics and new features, e.g., models and algorithms for failure propagation analysis. Third, XSAP implements a family of advanced routines for safety analysis that extend those of FSAP in many respects – namely, the BDD-based Fault Tree generation routines described in [15] are complemented by (different variants of) SAT-based and SMT-based routines, and routines based on IC3 [16,17]. Finally, while FSAP provides a graphical user interface, XSAP relies on an interaction shell similar to NUXMV, and models are expressed in textual format, thus increasing the flexibility and possibility of integration within other tools.

Some of the safety assessment functions of XSAP are used as a back-end for the COMPASS tool [5,18] and its extensions [19,20]. There are two key differences with respect to the COMPASS tools. First, XSAP provides a wider range of routines for Fault Tree generation; second, XSAP implements a general model extension mechanism, based on a library defining fault modes and their dynamics, while in COMPASS the fault models must be modeled manually and explicitly within the SLIM language.

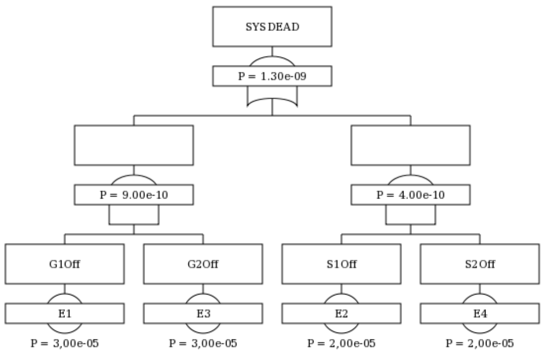
Other platforms for MBSA are based on the Altarica language and OCAS [21,22,23], on Scade [24,25], and on Statemate [26,27]. None of them is publicly available.

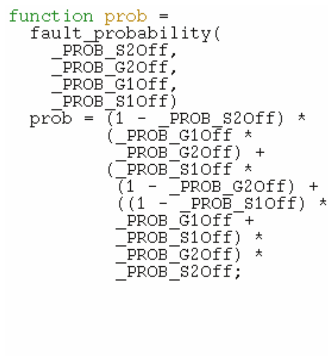
Finally, in [28] the authors present a framework for model based safety analysis, that provides transformations into different model checkers, including NuSMV, but does not provide specialized algorithms.

Structure of the paper. In Sect. 2 and 3 we describe the functionalities and the architecture of XSAP. In Sect. 4 we briefly discuss its most successful applications. In Sect. 5 we draw conclusions and outline future directions.

2 Functionality

In this section we describe the main features of XSAP. Fig. 1 illustrates the main flow.





```
function prob =
  fault_probability(
    _PROB_S2Off,
    _PROB_G2Off,
    _PROB_G1Off,
    _PROB_S1Off)
  prob = (1 - _PROB_S2Off) *
         (_PROB_G1Off *
          _PROB_G2Off) +
         (_PROB_S1Off *
          (1 - _PROB_G2Off) +
          ((1 - _PROB_S1Off) *
          _PROB_G1Off +
          _PROB_S1Off) *
          _PROB_G2Off) *
          _PROB_S2Off;
```

Fig. 2. An example FT and the associated symbolic probability.

2.1 Model Extension

Model extension [14,4] is an automated process that, based on a specification of the possible faults, returns a model (called *extended model*) that takes into account faulty behaviors. The model extension routine takes as input the *nominal model* (describing behavior in absence of faults), the *fault library* (containing templates for faults and their dynamics) and the *fault extension instructions* (specifying directives to instantiate the fault templates). Formal analyses can be run on the extended model, in order to assess system behavior in presence of faults.

The fault library of xSAP contains a comprehensive set of predefined fault modes, including, e.g., different variants of *stuck at*, *random*, *conditional*, *ramp down*, and can be further customized for any specific need. Moreover, a *local* and *global* dynamics libraries enable the definition of the dynamics of faults (e.g., *permanent* or *sporadic*). The fault library has been validated and extended to match the need of a significant case study of industrial size [13].

2.2 Safety Analysis

xSAP supports the automatic generation of artifacts that are typical of safety analysis, in particular Fault Trees and FMEA tables [29,6,7]. A Fault Tree (FT) is a graphical representation of the sets of possible causes of a given (undesired) event (the root of the tree – called *Top Level Event, TLE*). The TLE is linked by means of logical gates (AND, OR) to the basic events (faults). The minimal combinations of faults explaining the TLE are called *Minimal Cut Sets (MCSs)*. Fig. 2 (left) shows a sample FT for a battery sensor model: the FT links the TLE (a system failure) with basic events (generator and sensor faults). Finally, xSAP can generate Dynamic Fault Trees (DFTs) [30,31], where a *priority AND* gate is used to identify order of precedence of different events [32].

FMEA tables are a tabular representation of the causality relationships between (sets of) faults and a list of properties (representing undesired events, as in the case of FTs). xSAP also supports the generation of Dynamic FMEA tables, where order of events may be imposed.

2.3 Common Cause Analysis

Common Cause Analysis (CCA) is a necessary step of safety assessment, that is often required by safety standards, see e.g., [6,7]. It consists in evaluating the consequences of events that may break the hypothesis of independence of different faults. CCA aims at investigating possible dependencies, and evaluates the consequences in terms of system safety/reliability. xSAP enables the definition of events named *common causes*, which may trigger the occurrence of a set of (dependent) faults. Such faults may follow a user-specified pattern, e.g., *simultaneous* or *cascading* (subject to given temporal constraints). For instance, debris caused by an engine burst (the common cause) may cause multiple components of an aircraft to fail simultaneously. xSAP enables the evaluation of system reliability in presence of common causes and the generation of FTs including them.

2.4 Probabilistic Evaluation

xSAP supports the probabilistic evaluation of Fault Trees. Given numerical probabilities for the basic events and for the common causes, xSAP computes probabilities for the intermediate nodes and the TLE of a Fault Tree. With the exception of the constituent faults of common causes, all faults are assumed to be independent.

Furthermore, xSAP supports the symbolic computation of the formula representing the probability of the TLE, given the probability of the basic events. From such formula, xSAP produces the code in Python and Matlab/Octave. This can be used to sample the reliability of a system for different values of the fault probabilities, and plotted using visualization tools (e.g., Matlab [33]). Fig. 2 shows the FT with associated numerical probabilities, and the symbolic formula representing the probability of the TLE.

2.5 Failure Propagation Analysis

xSAP supports analysis of failure propagation using Timed Failure Propagation Graphs (TFPGs). A TFPG [34] is a graph-like model that accounts for the temporal progression of failures in dynamic systems and for the causality between failure effects, taking into consideration time delays, system reconfiguration and sensor failures. TFPGs may be seen as a more fine-grained model w.r.t. DFTs, and can be used to support important run-time activities such as diagnosis and prognosis [35,36,20].

The nodes of a TFPG represent either *failures* or *discrepancies* (representing anomalous behaviors). Edges represent propagation links; they are labeled with timing information (bounding the minimum and maximum propagation time) and modes (information on system modes enabling the propagation, e.g., operational modes of a spacecraft). Discrepancies may be given either AND or OR semantics – in the former case all incoming edges must be active in order for the failure to propagate, in the latter case any of them suffices. Fig. 3 shows a sample TFPG for the battery sensor model. Nodes without incoming edges (shown with dotted lines) are failures, whereas the remaining nodes are discrepancies (AND discrepancies are drawn as boxes, and OR discrepancies as circles). Edges are labeled with propagation bounds (in square brackets) and system modes (in curly brackets).

xSAP supports modeling and validation of TFPGs [37]. In particular, behavioral validation has the purpose to check whether a TFPG is a complete abstraction of a given system model (i.e., it contains at least as many behaviors as the system it represents).

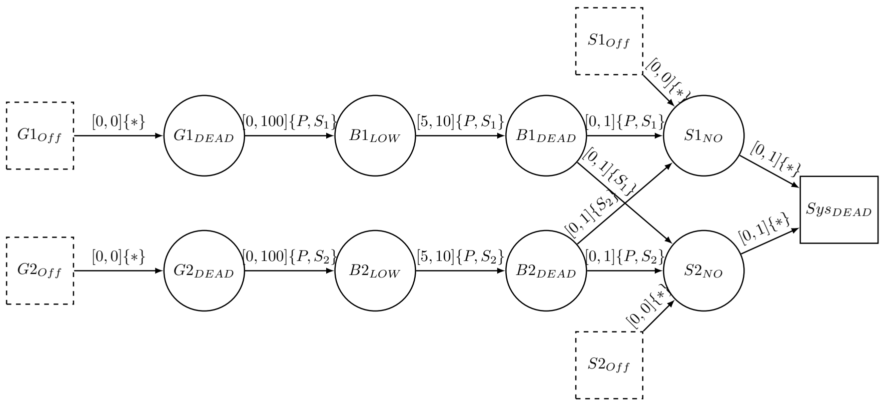


Fig. 3. An example TFPG.

Finally, XSAP supports automatic synthesis of a TFPG from a model, given a set of failures and discrepancies (currently, only the structure of the graph is synthesized). The integration of the TFPG validation features described in [38] is under way.

3 Architecture and Implementation

The architecture of XSAP is built around the NUXMV symbolic model checker [39], from which it inherits all the functionalities. NUXMV is an extension of NUSMV, and supports the verification of finite- and infinite-state systems, by means of advanced SAT- and SMT-based model checking techniques. NUXMV provides to XSAP the basic infrastructure, e.g., the symbol table, the flattening of the design, the Boolean encoding of scalar variables, the representation of state machines and temporal formulae, and the basic model checking algorithms.

On top of this, XSAP features the following blocks. *Model Extension* includes the library of fault modes, a parser for the fault extension instruction language, and the model extension. *Minimal cut sets computation* is realized by way of routines for parameterized model checking, using the model checking primitives of NUXMV as building blocks. *Fault Trees* can be generated/stored/retrieved either in XML or in a standard textual (tab-separated) format supported by commercial tools, such as FaultTree+ [40]. The management of *FMEA tables* is isolated in a separate module. Support for *Time Failure Propagation Graphs* is based on XML and textual formats. The textual format has been conceived to enable editing in a human-readable form – XSAP provides conversion from textual to XML and vice versa. *Syntax Directed Editors* (SDEs) are available for editing models, fault extension instructions, and TFPGs. Finally, the *Visualization* module contains graphical viewers that can be used to display the safety analysis artifacts: an FT Viewer and a TFPG viewer are available for the analysis of FTs and TFPGs, respectively.

xSAP has been developed in C and in C++ for the internal modules, while Python is used for model extension and TFPG manipulation. The viewers are based on the PyGTK, Goocanvas, PyGraphviz and Matplotlib libraries. xSAP compiles and executes on the most widely used Operating Systems (OSs) and architectures, namely: Linux, MS Windows, and MacOS X. Porting to other OSs is also possible (although not tested yet).

4 Applications

The xSAP platform has been used in a wide range of applications, both at academic and at industrial level, and in several domains, including avionics and aerospace, railway and industrial control. xSAP is the successor of FSAP [14] and retains all its features. FSAP has been developed within the ESACS [41], ISAAC [42], and MISSA [43] projects. It pioneered the ideas of model extension and model-based safety assessment, and was applied for safety assessment of avionics system [1,2,44].

xSAP has been widely used in several industrial projects with the European Space Agency (ESA). It is the back-end of the COMPASS tool [45,5,18], developed within the COMPASS [9], AUTOGEF [10,19], FAME [11,20] and HASDEL [12] projects funded by ESA.

Currently, xSAP is being used by Boeing [13]. The Boeing Company has evaluated xSAP in the context of a joint research and development project in the areas of model-based safety assessment, verification and validation, and contract-based design. The purpose of this project is to demonstrate the usefulness and suitability of model-based safety assessment techniques for improving the overall process in terms of robustness and cost-effectiveness, and for certification purposes. In this context, xSAP has been used to model an industrial-size case study [13,46] and thoroughly evaluated in an industrial setting.

5 Conclusions and Future Work

In this paper we presented xSAP, a state-of-the-art platform for model-based safety analysis, providing a full range of functionalities, based on symbolic model checking techniques. We described the architecture of xSAP and its industrial applications.

The symbolic technologies implemented in xSAP provide significant advances also in terms of scalability. We refer to [23] for a comparison with Altarica/OCAS (carried out using a license courtesy of Dassault Aviation), and to [17] for an exhaustive evaluation of the novel routines implemented in xSAP.

As future work, we intend to extend xSAP in several directions. First, we want to incorporate Contract-Based Safety Assessment (CBSA) techniques [47], enabling the generation of hierarchical FTs following the design structure. Moreover, we wish to incorporate the routines for evaluation of reliability architectures we developed in [48]. Finally, a significant extension will concern the definition of observability information in the model and the addition of related functionalities, such as diagnosability analysis and Fault Detection, Fault Isolation and Fault Recovery (FDIR) analysis [37].

References

1. Bozzano, M., et al.: ESACS: An Integrated Methodology for Design and Safety Analysis of Complex Systems. In: Proc. ESREL. (2003) 237–245
2. Bozzano, M., Cavallo, A., Cifaldi, M., Valacca, L., Villafiorita, A.: Improving Safety Assessment of Complex Systems: An Industrial Case Study. In: Proc. FME. Volume 2805 of LNCS. (2003) 208–222
3. Joshi, A., Miller, S., Whalen, M., Heimdahl, M.: A Proposal for Model-Based Safety Analysis. In: Proc. DASC, IEEE Computer Society (2005)
4. Bozzano, M., Villafiorita, A.: Design and Safety Assessment of Critical Systems. CRC Press (Taylor and Francis), an Auerbach Book (2010)
5. Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V., Noll, T., Roveri, M.: Safety, Dependability and Performance Analysis of Extended AADL Models. Computer Journal **54**(5) (2011) 754–775
6. SAE: ARP4754A Guidelines for Development of Civil Aircraft and Systems (December 2010)
7. SAE: ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (December 1996)
8. ECSS: European Cooperation on Space Standardization (Last retrieved on January 28, 2015) `http://www.ecss.nl`.
9. European Space Agency: ESTEC ITT AO/1-5458/07NL/JD "System-Software Co-Engineering: Performance and Verification" (2007)
10. European Space Agency: ESTEC ITT AO/1-6570/10/NL/LvH "Dependability Design Approach for Critical Flight Software" (2010)
11. European Space Agency: ESTEC ITT AO/1-6992/11/NL/JK "FDIR Development and Verification & Validation Process" (2011)
12. European Space Agency: ESTEC ITT AO/1-7263/12/NL/AK "Hardware-Software Dependability for Launchers" (2013)
13. Bozzano, M., Cimatti, A., Pires, A.F., Jones, D., Kimberly, G., Petri, T., Robinson, R., Tonetta, S.: A formal account of the AIR6110 Wheel Brake System. (2015) Submitted to CAV 2015.
14. Bozzano, M., Villafiorita, A.: The FSAP/NuSMV-SA Safety Analysis Platform. STTT **9**(1) (2007) 5–24
15. Bozzano, M., Cimatti, A., Tapparo, F.: Symbolic Fault Tree Analysis for Reactive Systems. In: Proc. ATVA. Volume 4762 of LNCS., Springer (2007) 162–176
16. Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: Ic3 modulo theories via implicit predicate abstraction. In: Proc. TACAS. (2014)
17. Bozzano, M., Cimatti, A., Mattarei, C., Griggio, A.: Efficient Anytime Techniques for Model-Based Safety Analysis. (2015) Submitted to CAV 2015.
18. Bozzano, M., Cimatti, A., Katoen, J.P., Katsaros, P., Mokos, K., Nguyen, V., Noll, T., Postma, B., Roveri, M.: Spacecraft Early Design Validation using Formal Methods. Reliability Engineering & System Safety **132** (2014) 20–35
19. 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. Volume ESA SP 701. (2012)
20. 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)
21. Bieber, P., Castel, C., Seguin, C.: Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System. In Grandoni, F., Thévenod-Fosse, P., eds.: Proc. EDCC-4. Volume 2485 of LNCS., Springer (2002) 19–31
22. Prosvirnova, T., Batteux, M., Brameret, P.A., Cherfi, A., Friedlhuber, T., Roussel, J.M., Rauzy, A.: The AltaRica 3.0 project for Model-Based Safety Assessment. In: Proc. DCDS. (2013)

23. Bozzano, M., Cimatti, A., Lisagor, O., Mattarei, C., Mover, S., Roveri, M., Tonetta, S.: Safety Assessment of AltaRica Models via Symbolic Model Checking. Science of Computer Programming **98**(4) (2015) 464483
24. Deneux, J., Åkerlund, O.: A Common Framework for Design and Safety Analyses using Formal Methods. In: Proc. PSAM7/ESREL. (2004)
25. Joshi, A., Heimdahl, M.: Model-Based Safety Analysis of Simulink Models Using SCADE Design Verifier. In Winther, R., Gran, B., Dahll, G., eds.: Proc. SAFECOMP. Volume 3688 of LNCS., Springer (2005) 122–135
26. Peikenkamp, T., Böede, E., Brückner, I., Spenke, H., Bretschneider, M., Holberg, H.J.: Model-based Safety Analysis of a Flap Control System. In: Proc. INCOSE. (2004)
27. Peikenkamp, T., Cavallo, A., Valacca, L., Böde, E., Pretzer, M., Hahn, E.M.: Towards a Unified Model-Based Safety Assessment. In: Proc. SAFECOMP. (2006) 275–288
28. Güdemann, M., Ortmeier, F.: A Framework for Qualitative and Quantitative Formal Model-Based Safety Analysis. In: Proc. HASE. (2010) 132–141
29. Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick III, J., Railsback, J.: Fault Tree Handbook with Aerospace Applications (2002)
30. Manian, R., Dugan, J.B., Coppit, D., Sullivan, K.J.: Combining Various Solution Techniques for Dynamic Fault Tree Analysis of Computer Systems. In: Proc. HASE, IEEE (1998) 21–28
31. Boudali, H., Crouzen, P., Stoelinga, M.: A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Sec. Comput. **7**(2) (2010) 128–143
32. Bozzano, M., Villafiorita, A.: Integrating Fault Tree Analysis with Event Ordering Information. In: Proc. ESREL. (2003) 247–254
33. Bozzano, M., Cimatti, A., Mattarei, C.: Automated analysis of reliability architectures. In: Proc. ICECCS, IEEE Computer Society (2013) 198–207
34. Karsai, G., Abdelwahed, S., Biswas, G.: Integrated diagnosis and control for hybrid dynamic systems. In: AIAA Guidance, Navigation and Control Conference. (2003)
35. Hayden, S., Oza, N., Mah, R., Mackey, R., Narasimhan, S., Karsai, G., Poll, S., Deb, S., Shirley, M.: Diagnostic Technology Evaluation Report For On-Board Crew Launch Vehicle. Technical Report TM-2006-214552, NASA (2006)
36. Ofsthun, S., Abdelwahed, S.: Practical applications of timed failure propagation graphs for vehicle diagnosis. In: Proc. Autotestcon, IEEE (2007) 250–259
37. Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal Design of Fault Detection and Identification Components Using Temporal Epistemic Logic. In: Proc. TACAS. Number 8413 in LNCS. Springer (2014) 46–61
38. Bozzano, M., Cimatti, A., Gario, M., Micheli, A.: SMT-based Validation of Timed Failure Propagation Graphs. In: Proc. AAAI. (2015)
39. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: Proc. CAV. (2014) 334–342
40. FaultTree+: FaultTree+ (Last retrieved on January 28, 2015) http://www.isograph.com/software.
41. ESACS: The ESACS Project (Last retrieved on January 28, 2015) http://www.transport-research.info/web/projects/project_details.cfm?ID=2658.
42. ISAAC: The ISAAC Project (Last retrieved on January 28, 2015) http://ec.europa.eu/research/transport/projects/items/isaac_en.htm.
43. MISSA: The MISSA Project (Last retrieved on January 28, 2015) http://www.missa-fp7.eu.
44. Bozzano, M., et al.: ISAAC, a Framework for Integrated Safety Analysis of Functional, Geometrical and Human Aspects. In: Proc. ERTS. (2006)
45. Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V.Y., Noll, T., Roveri, M., Wimmer, R.: A Model Checker for AADL. In Touili, T., Cook, B., Jackson, P., eds.: Proc. CAV. Volume 6174 of LNCS., Springer (2010) 562–565

46. SAE: AIR 6110, Contiguous Aircraft/ System Development Process Example (December 2011)
47. Bozzano, M., Cimatti, A., Mattarei, C., Tonetta, S.: Formal Safety Assessment via Contract-Based Design. In: Proc. ATVA. Number 8837 in LNCS. Springer (2014) 81–97
48. Bozzano, M., Cimatti, A., Mattarei, C.: Efficient Analysis of Reliability Architectures via Predicate Abstraction. In: Proc. HVC. Number 8244 in LNCS. Springer (2013) 279–294