The xSAP Safety Analysis Platform

Benjamin Bittner, Marco Bozzano(✉), Roberto Cavada, Alessandro Cimatti,
Marco Gario, Alberto Griggio, Cristian Mattarei, Andrea Micheli,
and Gianni Zampedri

Fondazione Bruno Kessler, Trento, Italy
bozzano@fbk.eu

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 and Failure Mode and Effects Analysis tables. Moreover, it supports probabilistic evaluation of Fault Trees, failure propagation analysis using Timed Failure Propagation Graphs, and Common Cause Analysis. 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) [1–3] 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 performed manually. Formal verification tools based on model checking have been extended to automate the generation of artifacts such as Fault Trees, which are required for certification of safety critical systems – see, e.g., [4,5].

xSAP is a platform for 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 funded by the European Space Agency. Moreover, xSAP is currently being used in a joint

M. Chechik and J.-F. Raskin (Eds.): TACAS 2016, LNCS 9636, pp. 533–539, 2016.
DOI: 10.1007/978-3-662-49674-9_31

research and development project between FBK and The Boeing Company [6]. xSAP is being developed by FBK, and it is currently distributed with a free license for academic research purposes and non-commercial applications. It can be downloaded from http://xsap.fbk.eu.

Related Work. xSAP is an evolution and a complete re-implementation of FSAP [7]. FSAP has been developed within the ESACS, ISAAC, and MISSA European projects. It pioneered the ideas of model extension and model-based safety assessment [2], and was applied for safety assessment of avionic systems. xSAP contains significant improvements, such as handling of infinite-state systems, more general and customizable libraries to define fault modes and their dynamics, and failure propagation analysis. Moreover, xSAP implements a family of novel routines for safety analysis: the BDD-based Fault Tree generation routines described in [8] are complemented by (different variants of) SAT-based and SMT-based routines, and routines based on IC3 [9].

Some of the safety assessment functions of xSAP are used as a back-end for the COMPASS tool [3,10] and its extensions, see e.g. [11]. 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 Altarica/OCAS [12–14], Scade [15,16], and Statemate [17]. They support a subset of the features included in xSAP (FTA, FMEA, or some limited form of model extension), but none of them is publicly available.

Structure of the Paper. In Sects. 2 and 3 we describe the functionality 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. Figure 1 illustrates the main flow.

2.1 Model Extension

Model extension [2,7] 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

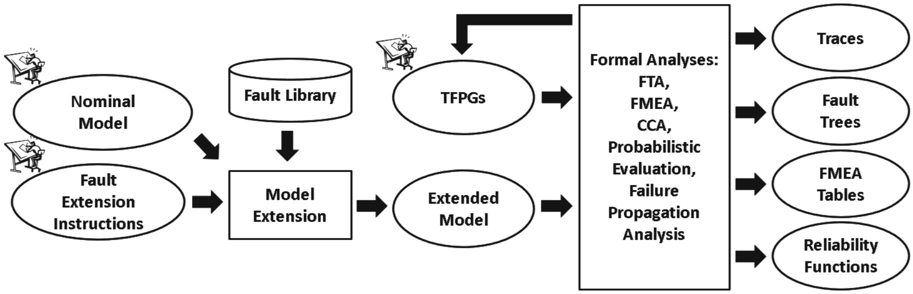
Nominal Model
Fault Extension Instructions
Fault Library
Model Extension
TFPGs
Extended Model
Formal Analyses: FTA, FMEA, CCA, Probabilistic Evaluation, Failure Propagation Analysis
Traces
Fault Trees
FMEA Tables
Reliability Functions

Fig. 1. The xSAP main flow.

of faults. The fault library of xSAP contains a comprehensive set of predefined fault modes, including, e.g., several 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 [6].

2.2 Safety Analysis

xSAP supports the automatic generation of artifacts that are typical of safety analysis, in particular Fault Trees and FMEA tables [4,18]. 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)*. Finally, xSAP can generate Dynamic Fault Trees (DFTs) [19], where a *priority AND* gate is used to identify order of precedence of events. FMEA tables are a tabular representation of the causality relationships between (sets of) faults and a list of properties (undesired events). xSAP also supports the generation of Dynamic FMEA tables, where the order of the 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 [4]. 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 reliability in presence of common causes and the generation of FTs including them.

2.4 Probabilistic Evaluation

xSAP supports 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 FT. With the exception of the constituent faults of common causes, all faults are assumed to be independent. Moreover, xSAP supports the computation in analytical form, as a Python or Matlab/Octave program, of the reliability function representing the probability of the TLE. Such programs can be used to sample the reliability function for different values of the probabilities, and to generate plots visualizing the TLE probability as a function of (a subset of) the parameters.

2.5 Failure Propagation Analysis

xSAP supports the analysis of failure propagation using Timed Failure Propagation Graphs (TFPGs) [20,21]. 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, system reconfiguration and sensor failures. TFPGs support important run-time activities such as diagnosis and prognosis [22]. The nodes of a TFPG represent either *failures* or *discrepancies* (representing anomalous behaviors). Edges represent propagation links, labeled with timing information (minimum and maximum propagation time) and modes (system modes enabling the propagation). 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.

xSAP supports modeling of TFPGs and the following analyses: validation of TFPG completeness (i.e., the TFPG contains at least as many behaviors as the system it represents) and tightness (i.e., parameters of the TFPG cannot be reduced without breaking its completeness). Moreover, xSAP implements a procedure for the automated synthesis of tight delay parameters for a given TFPG, and a procedure for the automated synthesis of the TFPG graph itself from a model, given a set of failures and discrepancies. Finally, xSAP integrates the TFPG validation features of [21].

3 Architecture and Implementation

The architecture of xSAP is built around the nuXmv symbolic model checker (http://nuxmv.fbk.eu). 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, model flattening, the Boolean encoding of scalar variables, the representation of state machines and temporal formulae, and the basic model checking algorithms. Moreover, xSAP relies on an interaction shell similar to the one of nuXmv, which increases the flexibility and possibility of integration within other tools.

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 [9], 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+. 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 enables 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 the graphical viewers: a trace viewer, an FT Viewer and a TFPG viewer are available for displaying and analyzing traces, 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.

4 Applications

The xSAP platform has been used in a wide range of applications, both industrial and academic, spanning several domains such as avionics and aerospace, railway and industrial control. xSAP has been widely used in several industrial projects with the European Space Agency (ESA), namely COMPASS, AUTOGEF, FAME and HASDEL (see http://es.fbk.eu/projects). It is the back-end of the COMPASS family of tools [3]. Finally, xSAP has also been used in a joint project with NASA [23].

Currently, xSAP is being used by Boeing [6]. 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. 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; xSAP has been used to model an industrial-size case study [6] 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 [14] for a comparison with Altarica/OCAS (carried out using a license courtesy of Dassault Aviation), and to [9] 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 [24], 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 [25]. 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 [20].

References

1. Joshi, A., Miller, S., Whalen, M., Heimdahl, M.: A proposal for model-based safety analysis. In: DASC. IEEE Computer Society (2005)
2. Bozzano, M., Villafiorita, A.: Design and Safety Assessment of Critical Systems. CRC Press (Taylor and Francis), an Auerbach Book, Boca Raton (2010)
3. Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V., Noll, T., Roveri, M.: Safety, dependability and performance analysis of extended AADL models. Comp. J. **54**(5), 754–775 (2011)
4. SAE: ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment., December 1996
5. ECSS: European Cooperation on Space Standardization. http://www.ecss.nl
6. Bozzano, M., Cimatti, A., Fernandes Pires, A., Jones, D., Kimberly, G., Petri, T., Robinson, R., Tonetta, S.: Formal design and safety analysis of AIR6110 wheel brake system. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 518–535. Springer, Heidelberg (2015)
7. Bozzano, M., Villafiorita, A.: The FSAP/NuSMV-SA safety analysis platform. STTT **9**(1), 5–24 (2007)
8. Bozzano, M., Cimatti, A., Tapparo, F.: Symbolic fault tree analysis for reactive systems. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 162–176. Springer, Heidelberg (2007)
9. Bozzano, M., Cimatti, A., Griggio, A., Mattarei, C.: Efficient anytime techniques for model-based safety analysis. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 603–621. Springer, Heidelberg (2015)
10. Bozzano, M., Cimatti, A., Katoen, J.P., Katsaros, P., Mokos, K., Nguyen, V., Noll, T., Postma, B., Roveri, M.: Spacecraf early design validation using formal methods. Reliab. Eng. Syst. Saf. **132**, 20–35 (2014)
11. Bittner, B., Bozzano, M., Cimatti, A., de Ferluc, R., Gario, M., Guiotto, A., Yushtein, Y.: An integrated process for FDIR design in aerospace. In: IMBSA (2014)
12. Bieber, P., Castel, C., Seguin, C.: Combination of fault tree analysis and model checking for safety assessment of complex system. In: Bondavalli, A., Thévenod-Fosse, P. (eds.) EDCC 2002. LNCS, vol. 2485, pp. 19–31. Springer, Heidelberg (2002)

13. 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: DCDS (2013)
14. Bozzano, M., Cimatti, A., Lisagor, O., Mattarei, C., Mover, S., Roveri, M., Tonetta, S.: Safety assessment of altarica models via symbolic model checking. Sci. Comput. Program. **98**(4), 464–483 (2015)
15. Deneux, J., Åkerlund, O.: A common framework for design and safety analyses using formal methods. In: PSAM7/ESREL (2004)
16. Joshi, A., Heimdahl, M.P.E.: Model-based safety analysis of simulink models using SCADE design verifier. In: Winther, R., Gran, B.A., Dahll, G. (eds.) SAFECOMP 2005. LNCS, vol. 3688, pp. 122–135. Springer, Heidelberg (2005)
17. Peikenkamp, T., Cavallo, A., Valacca, L., Böde, E., Pretzer, M., Hahn, E.M.: Towards a unified model-based safety assessment. In: Górski, J. (ed.) SAFECOMP 2006. LNCS, vol. 4166, pp. 275–288. Springer, Heidelberg (2006)
18. Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick III., J., Railsback, J.: Fault Tree Handbook with Aerospace Applications, NASA, Version 1.1. August 2002. http://www.hq.nasa.gov/office/codeq/doctree/fault_tree.htm
19. Manian, R., Dugan, J.B., Coppit, D., Sullivan, K.J.: Combining various solution techniques for dynamic fault tree analysis of computer systems. In: HASE, pp. 21–28, IEEE (1998)
20. Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal design of fault detection and identification components using temporal epistemic logic. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014 (ETAPS). LNCS, vol. 8413, pp. 326–340. Springer, Heidelberg (2014)
21. Bozzano, M., Cimatti, A., Gario, M., Micheli, A.: SMT-based validation of timed failure propagation graphs. In: AAAI (2015)
22. Abdelwahed, S., Karsai, G., Mahadevan, N., Ofsthun, S.: Practical implementation of diagnosis systems using timed failure propagation graph models. IEEE Trans. Instrum. Meas. **58**(2), 240–247 (2009)
23. Mattarei, C., Cimatti, A., Gario, M., Tonetta, S., Rozier, K.: Comparing different functional allocations in automated air traffic control design. In: FMCAD, pp. 112–119. IEEE (2015)
24. Bozzano, M., Cimatti, A., Mattarei, C., Tonetta, S.: Formal safety assessment via contract-based design. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 81–97. Springer, Heidelberg (2014)
25. Bozzano, M., Cimatti, A., Mattarei, C.: Efficient analysis of reliability architectures via predicate abstraction. In: Bertacco, V., Legay, A. (eds.) HVC 2013. LNCS, vol. 8244, pp. 279–294. Springer, Heidelberg (2013)