The FSAP/NuSMV-SA Safety Analysis Platform [⋆]

Marco Bozzano, Charles Jochim, and Francesco Tapparo

ITC-IRST, Via Sommarive 18, 38050 Povo, Trento, Italy
{bozzano,jochim,tapparo}@itc.it

1 The FSAP/NuSMV-SA Platform

FSAP/NuSMV-SA [1] consists of a graphical user interface (FSAP) and an engine (NuSMV-SA) based on the NuSMV model checker [2]. FSAP/NuSMV-SA is implemented in C++ as a cross-platform tool and it currently runs under Windows and Linux.

2 Description of the demo

The FSAP/NuSMV-SA platform aims to improve the development cycle of safety-critical systems, by providing a uniform environment that can be used both at design time and for safety assessment. Below we sketch the organization of the demo, by presenting a typical scenario of use of the platform, that can be illustrated by the steps in Fig. 1.

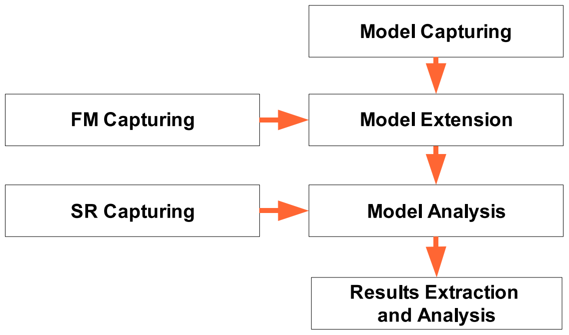

Fig. 1. Demo steps

2.1 Model Capturing

The starting point is a formal model of the system, called *system model*, written in some formal language. This model includes only the nominal behaviour of the system (that is, the system is assumed to be working as expected). This model is used by the design engineer to verify the functional requirements, and it is then passed to the safety engineer for safety assessment. In order to validate the system with respect to the safety requirements, the safety engineers enrich the behaviour of the system model by injecting failure modes, as described below.

2.2 Failure Mode Capturing and Model Extension

The second step includes the failure modes capturing and the model extension phases. The system model written by the design engineer must be extended by injecting the failure modes, that is, a specification of how the various components of the system can

⋆ This work has been supported by the E.U. ISAAC project, contract no. AST3-CT-2003-501848.

fail. This step yields a model that we call *extended system model*, in which all the components of the SM can fail according to the specified failure modes. The failure mode types to be injected into a system model can be stored and retrieved from a library of generic failure modes, called the *generic failure modes library*, and then automatically injected into the formal system model through an extension facility.

2.3 Safety Requirements Capturing

As long as a (extended) system model is available, it is possible to verify its behaviour with respect to the desired functional (nominal behaviour) and safety requirements (degraded behaviour). At this stage, design and safety engineers define the requirements that will be used to assess the behaviour of the system. The requirements can be either written using the temporal logic formula notation, or loaded from a library, called the *generic safety requirement library*, which provides a pattern-based definition for temporal formulas.

2.4 Model Analysis

At this stage, the behaviour of a system is assessed against the functional and safety requirements, by running the formal verification engine, that is, the NuSMV-SA model checker. Model analysis includes two main verification tasks. In the case of a system property, the model checking engine can test validity of the property, and generate a counterexample in case the system property is not verified. In case of a safety requirement, the model checking engine can be used to generate all possible minimal combinations of components failures, called *minimal cut sets*, that violate the safety requirements. Minimal cut sets can be arranged in the fault tree representation, a typical artifact of reliability analysis. They provide a convenient representation of the combination of events resulting in the violation of a given top level event, and are usually represented in a graphical way, as a parallel or sequential combination of AND/OR logical gates. The platform also supports the generation of another classical artifact of reliability analysis, that is, FMEA tables.

2.5 Result Extraction and Analysis

During this phase, the results produced by the model analysis phase are processed and presented in human-readable format. In particular, the result extraction phase is responsible for displaying all the outputs automatically generated by the model checking engine (e.g., simulation traces, fault trees, FMEA tables) and to present results of safety analyses in formats that are compatible with traditional fault tree analysis tools used by safety engineers. It is possible to visualize traces in textual, structured (XML), graphical, or tabular fashion. Fault trees can be viewed using commercial tools or using a displayer provided with the platform.

References

1. The FSAP/NuSMV-SA platform. `http://sra.itc.it/tools/FSAP`.
2. The NuSMV model checker. `http://nusmv.itc.it`.