Expressing best practices in (risk) analysis and testing of safety-critical systems using patterns

Herzner, Wolfgang; Sieverding, Sven; Kacimi, Omar; Böde, Eckard; Bauer, Thomas; Nielsen, Brian

Published in:
Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)

DOI (link to publication from Publisher):
10.1109/ISSREW.2014.24

Publication date:
2014

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Herzner, W., Sieverding, S., Kacimi, O., Böde, E., Bauer, T., & Nielsen, B. (2014). Expressing best practices in (risk) analysis and testing of safety-critical systems using patterns. In *Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 299-304). IEEE (Institute of Electrical and Electronics Engineers). https://doi.org/10.1109/ISSREW.2014.24

Expressing Best Practices in (Risk) Analysis and Testing of Safety-Critical Systems Using Patterns

Wolfgang Herzner
Safety & Security Department
AIT Austrian Institute of Technology
Vienna, Austria
Wolfgang.herzner@ait.ac.at

Sven Sieverding, Omar Kacimi, Eckard Böde
Transportation Department
OFFIS – Institute for Information Technology
Oldenburg, Germany
<sieverding,kacimi,boede>@offis.de

Thomas Bauer
Dep. of Embedded Systems Quality Assurance
Fraunhofer IESE
Kaiserslautern, Germany
thomas.bauer@iese.fraumhofer.de

Brian Nielsen
Department of Computer Science
Aalborg University, Denmark
bnielsen@cs.aau.dk

***Abstract*— The continuing pervasion of our society with safety-critical cyber-physical systems not only demands for adequate (risk) analysis, testing and verification techniques, it also generates growing experience on their use, which can be considered as important as the tools themselves for their efficient use. This paper introduces workflow patterns to describe such best practices in a systematic way that efficiently represents this knowledge, and also provides a way to relate different patterns, making them easier to identify and use, and cover as wide a range of experiences as possible. The value of the approach is demonstrated using some pattern examples from a collection developed in the Artemis-project MBAT[1]. Finally, the paper presents a wiki-based approach for developing and maintaining the pattern collection.**

Keywords—best practices; description patterns; A&T patterns; combining (risk) analysis and test

I. Introduction

As advanced safety-critical cyber-physical systems (CPS), such as those in the European transport domain (automotive, rail, and aerospace), increasingly pervade our society and daily lives, it becomes ever more pertinent to asses and reduce the risk of their deployment. At the same time, manufacturers are under a high pressure for delivering increasingly intelligent and feature rich products with short time-to-market, a high quality, and a low defect rate.

Advanced dynamic and static verification and validation (V&V) techniques such as model-based testing and simulation, static code analysis (e.g., based on abstract interpretation), formal model-analysis (e.g., based on model-checking) used with integrated risk- and safety analysis can be used to target these challenges. Risk analysis is actually a crucial part of engineering and V&V of complex safety critical systems as risk can help prioritizing effort. Development of any complex product will the demand application of a multitude and mixture of these techniques, so it also becomes very important to know *how* these are to be methodologically used and combined, preferably in optimizing configurations, that reflects effective solutions and best practices.

This paper proposes a method called *analysis- and test patterns* (A&T patterns) for describing typical and efficient workflow integrated risk and safety analysis, dynamic test and static analysis for V&V of CPS. By collecting, describing, and inter-relating the A&T patterns there is a great potential for capturing best practices and valuable experiences for advanced V&V solutions, and for communicating such ideas among researches and industrial engineers, with the ambition of advancing state-of the art in CPS and interoperable tooling.

Fig. 1 depicts this overall idea of integrated risk analysis, test and analysis.

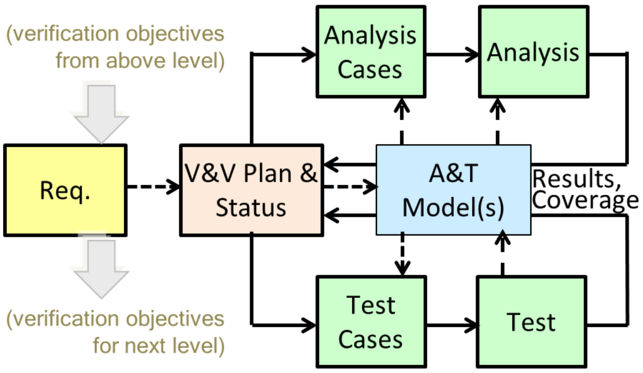

Fig. 1 Overall analysis and test (combination) method

Briefly stated, the engineers develop an initial V&V plan defining how and by what techniques each requirement for the considered component (and abstraction level) is addressed.

[1] MBAT (Combined Model-based Analysis and Testing of Embedded Systems) is a European industry led applied research project under Artemis Grant #269335, http://www.mbat-artemis.eu/home/, which partially funded this work

This includes an initial risk[2] assessment that will guide the required effort and choice of V&V techniques: Requirements and components with less risk call for use of less powerful and less demanding techniques; conversely with more critical requirements. Based on this initial plan, the engineers define and execute a number of analysis and test cases that results in a set of verdicts (pass, fail, inconclusive), measures of coverage, confidence, complexity etc., and possible indications of suspect behavior that require further investigation. These results are then systematically collected and fed back into the V&V planning activity, including re-assessment of risk, and are exploited to create a revised, optimized plan. This idea of combining different analysis and testing techniques can be used in several ways. For example, if a test reveals a defect, it may be worth the effort to target the problematic component with analysis due to the bug-clustering assumption. Similarly, if model analysis identifies a defect, it may be worthwhile to create additional test cases for that objective to increase confidence (and reduce risk) of the implementation. For the purpose of *coverage-completion* of a test suite with respect to a given test-criteria (e.g., branch or MC/DC coverage), insufficient coverage may be improved by utilizing a model-checker (or static code-path synthesis tool [6]) to synthesize the test cases for the missing coverage items [1] by interpreting the counter example (or path condition) as a test case.

Additional descriptions of the overall methodology are provided in [7]. However, this generic overall method may be used in several different settings. To make the solutions and concrete application more clear and usable, we developed an approach for defining workflow patterns that allows describing best practice solutions in a harmonized way, and which brings the individual workflows into relation with each other.

After addressing related work, the paper introduces the principles of A&T patterns along with a systematic method for describing them. Two concrete examples of applying model-based safety analysis (from a collection developed in the Artemis-project MBAT) demonstrate the value of the approach. This is followed by description of a wiki-based approach for presenting the A&T patterns catalogue developed in MBAT. The paper closes with a summary and outlook.

II. Related Work

Based on earlier works in the domain of architecture [8], the concept of *design patterns* for software engineering became very popular, as they allowed describe working solutions for dedicated software development problems, in particular exploiting the new paradigm of object-orientation. The presumably most popular book was [9], but a real massive publication activity occurred at that time, so that already in 2000 a "Pattern Almanach" could be published with more than 900 entries [12].

While software engineering is a central application field, the principle of *solution patterns* can be used in presumably any problem domain, since "A (solution) **pattern** is a (generic) **solution** for a certain **problem** in a given **context**" [13]. This means it is not a fixed rule but a principal description or template for how to solve a problem. A good pattern defines best practice that can be re-used in many different situations. So, patterns have been described for a number of different domains, for example in education[3] [2], organization[4,5], or management[6][10]. The related paper [11] presents a method for defining patterns and pattern evaluation criteria.

For analysis and test, a number of pattern collections exist. But they deal with requirements analysis for deriving software design[7] or with test automation, e.g. [8] and [4]. To our best knowledge, the combination of model-based analysis and test methods has not been described using pattern principles so far.

III. A&T Patterns Principles

A. Sources and Scope

The generic A&T methodology outlined in the introduction was applied in the project use cases, which consisted of a number of grouped tasks (called *scenarios*). It turned out that several processing steps taken to implement these tasks prove more successful than others, but also showed similarities, i.e. common work-flow patterns behind them. This motivated to develop a description technique that allows recording these experiences such that not only their reuse is fostered but also that as much as additional information is efficiently captured.

B. A&T Pattern Items

Several pattern notations have been proposed and used in literature, see section II for references. Given the observation that A&T patterns are essentially work-flows and that we wanted to collect additional information systematically, we chose following description items.

Name: A good name is essential. It should tell the problem or context and indicate, if possible, the solution idea such that a reader can quickly guess whether a pattern can help for a given problem.

Purpose: A concise description of the addressed problem, perhaps outlining the proposed solution approach.

Context/Pre-conditions: Characterization of the situations where the pattern can be applied.

To consider: Aspects to be considered when applying the pattern; they can help to decide about precise pattern application. Examples are required level of user knowledge or costs of involved tools.

Structure: This is the core of the pattern. A&T patterns are workflows, and these are illustrated by workflow diagrams for which predefined graphical symbols are used, preferably based on standardized notations like Business Process Model and Notation (BPMN) or UML Activity Diagrams, that are

[2] In this paper, risk is defined generically as the probability of occurrence of a failure times a metric of its severity.

[3] http://en.wikipedia.org/wiki/Pedagogical_patterns

[4] http://en.wikipedia.org/wiki/Organizational_patterns

[5] http://www.fearlesschangepatterns.com/

[6] http://crinfo.univ-paris1.fr/EKD-CMMRoadMap/index.html

[7] http://en.wikipedia.org/wiki/Software_analysis_pattern

[8] http://msdn.microsoft.com/en-us/library/cc514239.aspx#CommonTestPatterns_topic2

understandable by a wider audience and enable tool support. See section IV for examples.

Participants: A description of entities used in the pattern and their roles. Examples are humans, tools, and data.

Actions/Collaborations: A description of what active entities do and how entities used in the pattern interact with each other.

Discussion: Further comments and possible consequences can also be described. This often includes practical application hints as well as potential limitations.

Known uses: Examples how the pattern is applied in practice.

Related Patterns: A listing of other A&T patterns that complement the described pattern or have other relationships to it.

IV. A&T Pattern Examples

In this section, we give two examples of A&T patterns and their intention, using the notation introduced in Section III.

A. *MBSA with Fault Injection*

Purpose: Given the increasing complexity of safety-critical systems, it is more and more difficult to perform the safety assessment required by standards. The following pattern presents how to perform a so called Model-Based Safety Analysis (MBSA) [14]. Its results can be used for classical safety assessment techniques such as Fault Tree Analysis (FTA) and Failure Modes and Effect Analysis (FMEA).

The MBSA has two main prerequisites. The first is the use of formal semantics to express the analyzed malfunctions as well as safety requirements. This enables fault to be injected in the nominal system model and extending it to include the behavior in case of fault occurrences. Additionally, formal safety requirements can be translated into failure observers. The second prerequisite is the underlying verification technique that analyzes whether the failure of the safety requirement can be observed in the extended system model. A typical result of a traditional safety analysis, that can be generated by the MBSA, is the minimal cut-sets (MCS) for a given safety requirement. Cut-sets are sets of fault combinations that cause the safety requirement to fail. Specifically, if any fault is removed from a minimal cut-set, the remaining set is no longer a cut-set.

To consider: Implementing this pattern requires the involvement of a safety engineer. Also, the execution of the MBSA might be fairly time and resources consuming.

Structure: See **Fig. 2**. Dashed borders of activity boxes indicate substructure, which cannot be shown here due to space limitations. See the pattern wiki (section V) for more details.

Actions/Collaborations:

1. Formal requirements are created on the basis of natural-language requirements.
2. The system implementation model is created based on the formal requirements. Even if the workflow could start alternatively with natural language requirements, emphasis is put on formalized requirements to avoid the semantic ambiguity of natural language.
3. Extending the nominal system model with malfunction behavior using fault injection techniques, resulting in an extended system model.
4. A failure observer of the analyzed safety requirement(s) is generated. This observer automaton will be used later at the level of the analysis to monitor whether the system has reached a failure state.

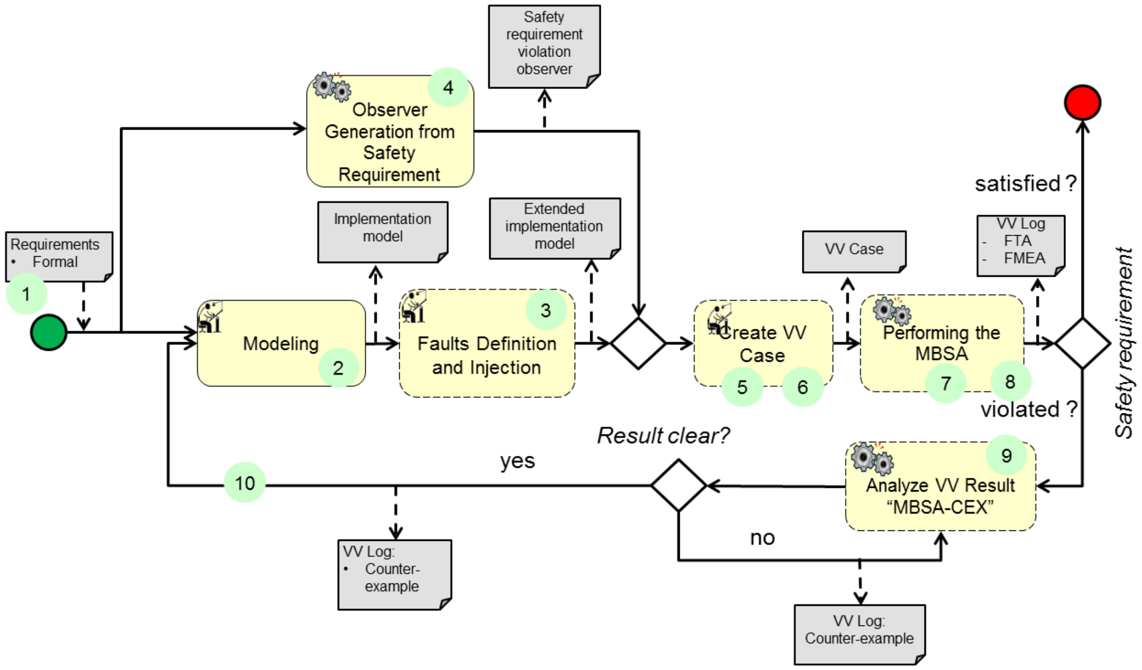

Fig. 2 Workflow of AT-pattern "MBSA with Fault Injection" (circled numbers refer to activities explained under Actions/Collaborations)

5. A safety analysis task is created and linked to the extended model and the failure observer.
6. The type of the safety analysis task is chosen, which can be either FMEA or FTA.
7. The model-based safety analysis is executed.
8. Finally, either a fault tree or an FMEA table is generated depending on the type of the analysis chosen.
9. The safety engineer assesses the results and the outcomes for safety requirements violation (caused by the injected failure-modes). He can run a counter example (CEX) analysis to obtain a trace that shows how the faults lead to the violation of the requirement.
10. The system engineer will modify the design according to the analysis results by introducing additional safety mechanisms.

Comments: MCS can be used to automatically derive artifacts such as fault trees and FMEA tables. In some cases, the information contained in the generated safety artifacts is not trivial i.e. it is not clear how the occurrence of the malfunctions lead to the violation of the requirement. Accordingly, techniques such as model-checking can be used to generate a counter example (CEX) corresponding to a simulation run that clearly shows how the faults lead to a situation where the safety requirement is violated.

Known Uses:

The Model-Based Safety Analysis framework from OFFIS is an instantiation of this pattern. This framework comprises a set of tools, e.g., the OFFIS PatternEditor to formalize requirements, some internal tools to define the Failure Modes and the VV Cases, as well as the VIS model checker[9] to perform the analysis. In order to make the tools more interoperable, e.g., use them in instantiations of other AT patterns, all of these tools are connected via an OSLC[10] compliant interface. Within the MBAT project, the MBSA successfully demonstrated the applicability of the approach to evaluate a prototypical Simulink/Stateflow implementation model of a turn indicator system.

The overall setup and the application of the workflow of the pattern in the use-case are presented more in detail in [19].

B. Support Safety Analysis by Fault Simulation

After running the model-based safety analysis on an implementation model, a fault combination leading to a system failure is generated. This CEX is the starting point for this A&T pattern. The workflow is depicted in **Fig. 3**.

The MBSA should normally be run again to assess whether the safety mechanism mitigates the given fault combination. Since the MBSA can be quite time costly to run, the pattern proposes to make use of the generated counter-example. The pattern suggests performing a simulation of the counter-example generated by the MBSA on a system model injected with faults. Hence, if the modifications introduced to the system are not sufficient for reaching the safety goal, this can be discovered prior to running the MBSA another time. The workflow of the pattern is described in the following:

Actions/Collaborations:

1. The engineering models are created on the basis of natural language requirements.
2. The MBSA workflow is performed and the outcomes of the analysis are generated.
3. The safety engineer assesses the V&V Log for safety requirements violation (caused by the injected *faulty/dysfunctional behavior).*

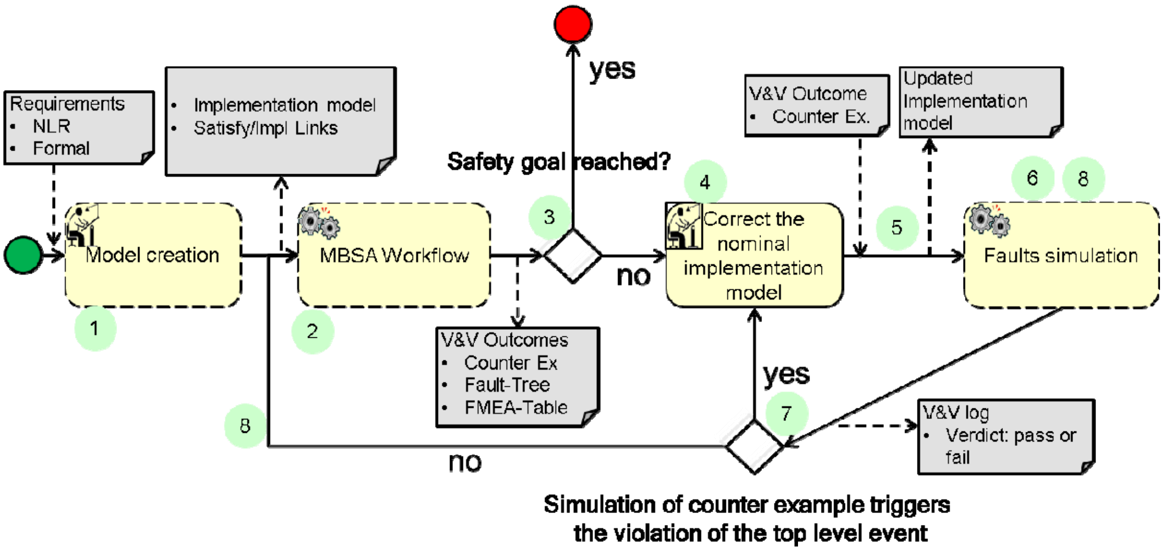

Fig. 3 Workflow of AT-pattern "Support Safety Analysis by Fault Simulation" (circled numbers refer to activities explained in pattern description)

[9] http://vlsi.colorado.edu/~vis/

[10] http://open-services.net/

4. The system designer will modify the system according to the analysis results, e.g., by introducing new safety mechanisms.
5. Faults are extracted from the counter-examples generated by the MBSA and are injected in the system model.
6. A fault-simulation of the extended system model is performed.
7. If the fault-simulation shows that the introduced changes are not effective and the top level event can still be triggered by the CEX, step 4 needs to be repeated.
8. The fault-simulation is performed again, if it results in a failure start from 7 again. Otherwise, the test is successful and the MBSA can be run on the updated system.

V. The A&T Pattern Catalogue

In order to establish and support a living quality assurance pattern community, we implemented a wiki-based approach for presenting the A&T pattern developed in MBAT. The pattern wiki [15] currently contains 16 patterns. It enables the quick and easy addition and modification of patterns including the support of editable diagrams for representing the solution workflows.

Hosting of the web server and quality assurance of submissions is currently done by the leaders of the MBAT technology work package. Until now, the pattern wiki has helped to launch an active MBAT pattern community with a core set of contributing authors and a large readership within the project. Broader announcement of the resource for external interested parties has started.

For our system, we use the MediaWiki platform [16]. The implementation of the MBAT pattern wiki consists of an overview page, which provides easy access to all pattern-specific pages. Each pattern page contains links to related entities to facilitate navigation in the pattern set.

One basic requirement for the technical implementation was quick and easy support for editable diagrams, especially for the pattern relationships in the overview figure and the concrete workflow descriptions in the pattern-specific pages. In the end, two types of diagrams are used: the pattern set and its relations (i.e. the "overview page") is generated using a simple state machine diagram, and UML 2.x activity diagrams to describe the workflow diagrams in every pattern. Several libraries were evaluated. Considering the maturity and usability of the solutions, their integration into the MediaWiki system, and the ability to generate compact diagrams, PlantUML (for the workflow diagrams [17], which allows automated generation of these diagrams from textual representations) and GraphViz (for the overview diagram [18]) were selected.

Fig. 4 shows a presentation of the MBAT pattern collection that serves as the main page in the wiki. The figure was created by the GraphViz library. Each pattern is represented as a node. The edges represent the connections and main relations between different patterns. Edge labels provide additional comments on the concrete relationships. The wiki pages of the single patterns are linked to the corresponding nodes in the overview figure via hyperlinks.

Fig. 5 shows an example of such an automatically generated workflow diagram. The GraphViz library has some limitations with respect to our needs. For example, multiple activity labels or comments connected to arcs are not supported, and the empty diamond at bottom is needed to close branching.

Till the current status of the MBAT project, the wiki has been a valuable source for getting in touch with the MBAT

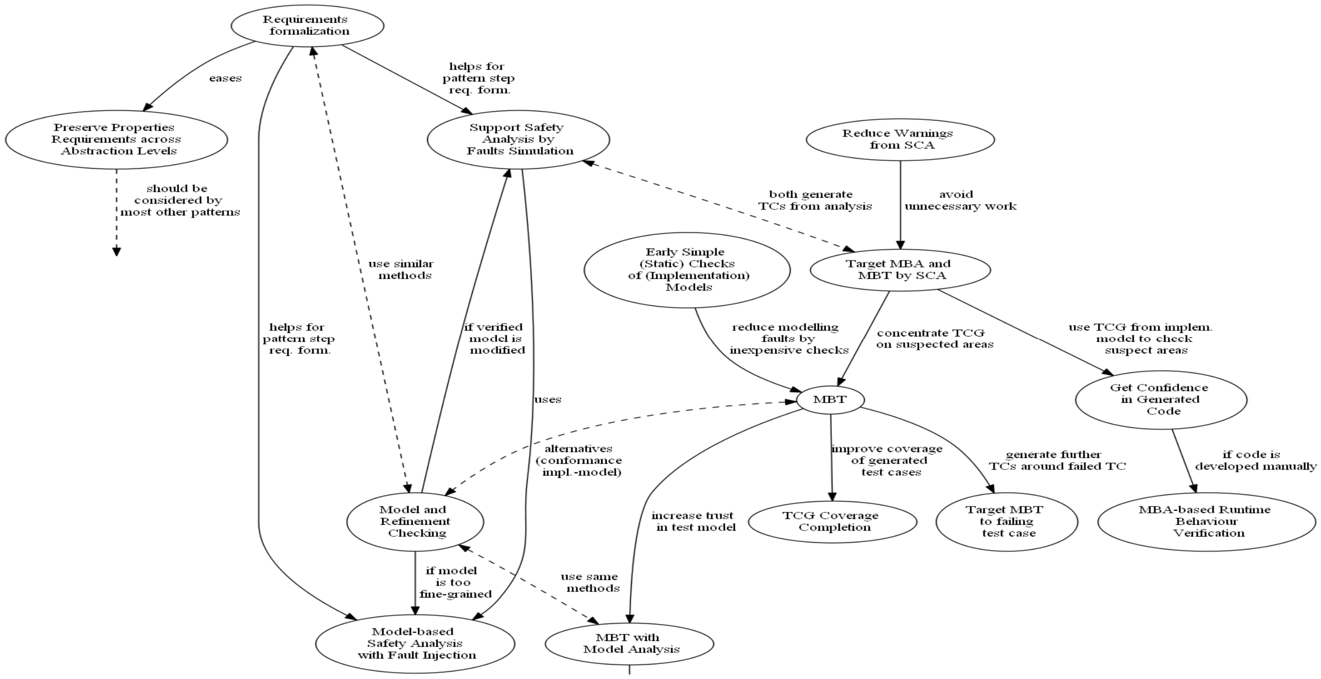
Requirements formalization
eases
helps for pattern step req. form.
Preserve Properties Requirements across Abstraction Levels
Support Safety Analysis by Faults Simulation
Reduce Warnings from SCA
should be considered by most other patterns
both generate TCs from analysis
avoid unnecessary work
Early Simple (Static) Checks of (Implementation) Models
Target MBA and MBT by SCA
use similar methods
helps for pattern step req. form.
if verified model is modified
reduce modelling faults by inexpensive checks
concentrate TCG on suspected areas
use TCG from implem. model to check suspect areas
uses
MBT
Get Confidence in Generated Code
alternatives (conformance impl.-model)
improve coverage of generated test cases
generate further TCs around failed TC
if code is developed manually
Model and Refinement Checking
increase trust in test model
TCG Coverage Completion
Target MBT to failing test case
MBA-based Runtime Behaviour Verification
if model is too fine-grained
use same methods
Model-based Safety Analysis with Fault Injection
MBT with Model Analysis

Fig. 4 MBAT A+T pattern system (pattern relationship diagram)

patterns and quality assurance solutions. More detailed information on the pattern is provided by partner and case study specific documents. Public documents are linked to the pattern pages. For a more serious discussion, the corresponding experts and contact persons are listed.

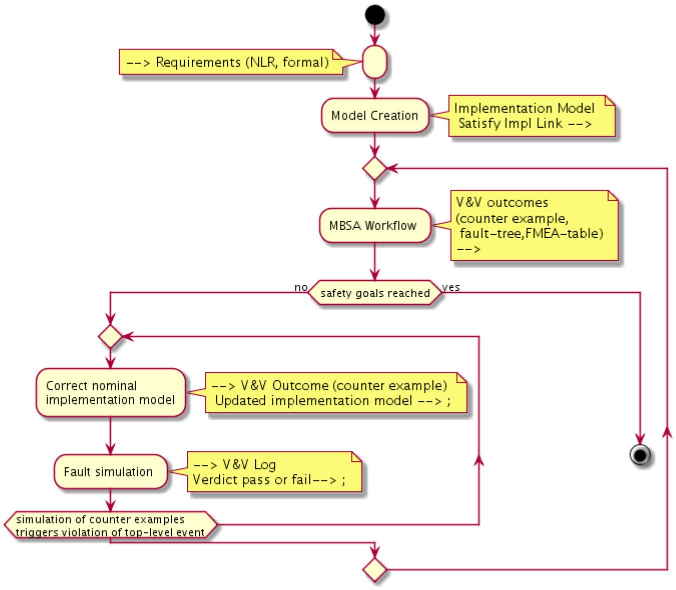

Fig. 5 Activity diagram of the ”Support Safety Analysis by Fault Simulation” workflow

VI. CONCLUSION

A. *Summary*

This paper presents an approach for describing best practices on integrating (risk/safety) analysis and testing in a systematic manner by means of so called A&T patterns, which are developed in the Artemis-project MBAT. The approach is adapting the principle of solution patterns used widely for software engineering. An A&T pattern is a dedicated workflow for combining analysis and test steps for meeting complex requirements, improving efficiency, saving effort, etc. Besides the workflow diagram, the description of an A&T pattern contains additional information that not only helps to apply the pattern effectively, but also to quickly assess whether a certain pattern can help to solve or mitigate a certain problem. A few examples illustrate these concepts.

A further significant benefit comes with the combination of A&T patterns. For that purpose, relationships between patterns are described, which can be illustrated in a pattern relationship diagram. Such a diagram is used as entry page for a public web presentation of the A&T patterns developed during the MBAT project.

B. *Outlook*

We envision two major directions of progressing work on the A&T patterns. One is extending the A&T patterns catalogue (by describing new patterns), which will also be guided by trying to fill gaps in the pattern collection where possible, and improving of existing descriptions. This includes identification of further pattern applications (known uses).

The other is hosting and maintaining the public pattern wiki described in the previous section. This will include improvement of diagram presentations, for example to add activity and collaboration labels to workflow diagrams.

It is hoped that the A&T patterns work started in MBAT will not only survive the project end, but also become a viable means for exchanging best practices in the V&V community.

REFERENCES

[1] T. Blackmore, D. Halliwell, P. Barker, K. Eder, and N. Ramaram, “Analysing and closing simulation coverage by automatic generation and verification of formal properties from coverage reports”, in Integrated Formal Methods 2012, pages 84–98. Springer

[2] J. Bergin. “Pedagogical Patterns: Advice For Educators”, Joseph Bergin Software Tools, 2012, ISBN: 0985154381

[3] J. Coplien, “Organizational Patterns”, in J. Coplien and D. Schmidt, eds., Pattern Languages of Program Design. Addison-Wesley, 1995, pp. 183 – 237

[4] A.-G. Vouffo Feudjio, I. Schieferdecker, “Test Automation Design Patterns for Reactive Software Systems”, ceur-ws.org/Vol-566/E6_BlackBoxTesting.pdf

[5] M. Fowler, “Analysis Patterns: Reusable Object Models”, Addison-Wesley. 1996, ISBN 0-201-89542-0

[6] J.C. King, “Symbolic execution and program testing”; Commun. ACM, 19(7):385–394. 1976

[7] B. Nielsen, “Towards a Method for Combined Model-based Testing and Analysis”, in Proc. 2nd International Conference on Model-Driven Engineering and Software Development, pp. 609. January 2014

[8] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, “A Pattern Language”, Oxford University Press, 1977

[9] E. Gamma, R. Helm, R. Johnsohn, and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Systems”, Addison Wesley, 1995

[10] C. Rolland, S. Nurcan, G. Grosz, “Enterprise knowledge development: the process view”, Information & Mgmt 36, Elsevier 1999, 165-184

[11] R.S. Kennett, “Implementing SCRUM using Business Process Mgmt and Pattern Analysis methodologies, in Dynamic Relationship Mgmt Journal”, Nov.2103, http://www.sam-d.si/Upload/Articles/DRMJv02n02a03.pdf

[12] L. Rising, “The Pattern Almanac 2000”, Addison-Wesley, 2000

[13] N.B. Harrison, B. Foote, and H.Rohnert, (eds.), “Pattern Languages of Program Design 4”, Addison-Wesly, 2000 (Proceedings of PLoP’98, EuroPLoP’98, or earlier (Euro)PloP-conf.s)

[14] T. Peikenkamp, A. Cavallo, L. Valacca, E. Böde, M. Pretzer, E.M. Hahn, “Towards a unified model-based safety assessment”, in: Proceedings of SAFECOMP. pp. 275–288 (2006)

[15] MBAT pattern wiki [retrieved on 11 Aug 2014] http://mbat-wiki.iese.fraunhofer.de

[16] MediaWiki website [retrieved on 11 Aug 2014] https://www.mediawiki.org/wiki/MediaWiki

[17] PlantUML website [retrieved on 11 Aug 2014] http://plantuml.sourceforge.net/

[18] GraphViz website [retrieved on 11 Aug 2014] http://www.graphviz.org

[19] O. Kacimi, C. Ellen., M. Oertel., D. Sojka, “Creating a reference technology platform: Performing model-based safety analysis in an heterogeneous development environment”, in Proceedings of the MODELSWARD Conference (2014)