Standards-based Assessment of Development Toolchains in Safety-Critical Systems

Zoltán Szatmári

Standards-based Assessment of Development Toolchains in Safety-Critical Systems

Zoltán Szatmári
Budapest University of Technology and Economics
Email: szatmari@mit.bme.hu

***Abstract*—To reduce the risks of software design failures, the software development processes are more and more subject to regulations fixed in (domain-specific) standards that define criteria for the selection of techniques and measures. In this paper we propose a method for the assessment of development processes and toolchains. The tasks and tools in the development process are modelled and then classified using an ontology that is constructed on the basis of the standard, and a reasoning tool is applied to check whether the criteria are satisfied.**

I. Introduction

Our everyday life depends on software to a considerable extent, this way the reduction of the risks of design and implementation failures is of utmost importance. The software development processes are more and more subject to regulations fixed in (domain-specific) standards that define criteria for the selection of proper development methods and measures. Accordingly, if software is deployed in a critical environment then an independent assessment is needed to certify that its development process is compliant to the criteria stated in the related standard. The goal of our work is to support the assessment of development processes and toolchains by elaborating a formal verification technique that allows the automated checking of the compliance to standards. On the analogy of classical model checking (that is applied to examine whether a formal design model satisfies some temporal requirements) we represent the development process and tools in a *process model* and use a reasoning tool to check whether the criteria originated from the standard are satisfied.

This vision necessitates the solution of the following tasks:

- Formalisation of the requirements (criteria) in standards that concern the selection of methods and tools.
- Definition (or adaptation) of modeling techniques to describe the relation of methods, the capabilities of tools, and the construction of (domain-specific) development processes.
- Elaboration of techniques that check the compliance of concrete development processes (constructed by process designers) to the requirements.

The formalisation of the requirements and the model-based description of tools and methods open a way to support also the synthesis of processes and toolchains that are compliant to the standard. The process designer can be assisted by

- identifying missing methods and tools,
- proposing alternative solutions,
- offering a library of toolchain patterns,
- optimizing processes from the point of view of costs, time, safety etc.

In the following we describe ideas and initial results related to the implementation of the above mentioned tasks and present a simple example.

II. Formalisation of the requirements

Formalisation is a prerequisite of both formal verification and synthesis support. We focused on the development processes for safety critical applications, and analyzed the EN50128 standard for railway applications [1]. This standard defines five safety integrity levels (SIL) for development processes and describes methods that can be applied during the process. For each development step the mandatory (M), highly recommended (HR), recommended (R) and not recommended (NR) methods are described in a tabular form.

The main challenges during the requirement formalisation are the following:

- The development methods are refined hierarchically, i.e., several high level methods are decomposed into alternative combinations of lower level ones (see Fig. 1).
- For each SIL different requirements are described in the standard. Accordingly, this introduces a new dimension into the requirement formalisation.
- The sufficient conditions for every SIL are formulated in the standard using various combinations of the applied methods.

Technique/Method	SIL1	SIL2	SIL3	SIL4
1. Formal Proof	R	R	HR	HR
2. Probabilistic Testing	R	R	HR	HR
3. Static Analysis	HR	HR	HR	HR
4. Dynamic Analysis and Testing	HR	HR	HR	HR
5. Metrics	R	R	R	R
6. Traceability Matrix	R	R	HR	HR
7. SW Error Effect Analysis	R	R	HR	HR

Fig. 2. The Verification and Testing methods (EN50128)

In the following, the *Verification and Testing* step of the development process described in the EN50128 standard is presented as a small example in order to demonstrate the mentioned concepts.

In Fig. 2 the recommendation level of some methods is shown and the hierarchy of the methods is depicted in Fig. 1. The combination of the required methods are expressed as

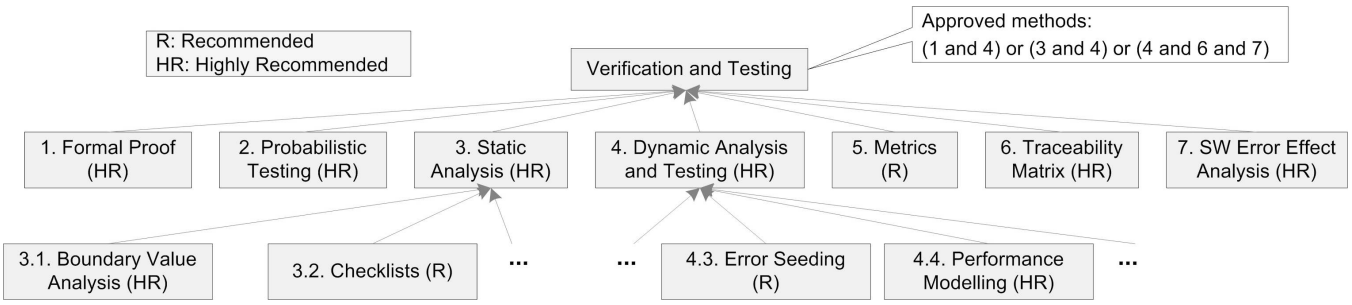

Fig. 1. Verification and Testing methods for SIL4 (EN50128)

follows: *„For Software Integrity Levels 3 and 4, the approved combinations of techniques shall be (1 and 3) or (3 and 4) or (4, 6 and 7)"*

A formal representation of the hierarchical structure of methods can be provided by defining an ontology [3] and describing it using description logic. Here concepts refer to the development methods and their relations include the refinement.

An ontology definition consists of two types of statements:

- The T-Box (Terminology Box) statements are used to describe the domain specific context in terms. Concepts of the domain, those attributes and relations between them are defined.
- The A-Box (Assertion Box) consists of T-Box-compliant statements. These describe the individuals, i.e. the instances of the concepts defined in the T-Box.

Based on this standard the methods ontology can be constructed. The hierarchy of methods can be expressed using concept hierarchy in the ontology T-Box (Terminology box). In Fig. 3 part of the formal ontology description is shown for the *Verification and Testing* step.

$$\begin{aligned} StaticAnalysis &\sqsubseteq Verification\_And\_Testing \\ Checklists &\sqsubseteq StaticAnalysis \\ Boundary\_Value\_Analysis &\sqsubseteq StaticAnalysis \\ Control\_Flow\_Analysis &\sqsubseteq StaticAnalysis \end{aligned}$$

Fig. 3. Formal ontology description

In Fig. 4 part of the graphical representation of the ontology is depicted that is constructed using the Protege ontology modeling tool [6].

III. Assembling an extensible tool repository

The next step of the formalisation process is the construction of the tool repository. This repository is a collection of tools that can be used in a given company during the (construction of the) development processes. Each available tool is classified on the basis of the concepts defined in the ontology constructed in the previous step, i.e., for each tool the supported methods are given.

The following items can be found in this repository:

- *Simple tools*, that realize a specific method described in the specification. For example, the SPIN model checker [14] and the PVS theorem prover can be classified as tools supporting Formal Proof. The PolySpace can be classified as a tool supporting Symbolic Execution which is a Static Analysis method.
- *Toolchain patterns* that are formed by tools that have to be executed in a predefined sequence, to support a given method. For example, Structure-Based Testing (which is a Dynamic Analysis and Testing method) can be supported by the following toolchain:
  1) Model transformation from UML2 statechart model to the input format of the SAL model checker.
  2) Test generation for a given coverage criteria using the SAL-ATG [13] tool.
  3) Mapping abstract test cases to executable test cases.
  4) Execution of test cases and measuring coverage by Rational RealTime.
- *Abstract development steps* that are also allowed in order to support high level design of development processes. For example, „model checking of the reachability of hazardous states" can be classified as Formal Proof. In the next refinement steps tools like SPIN or SAL can be assigned to this abstract step.

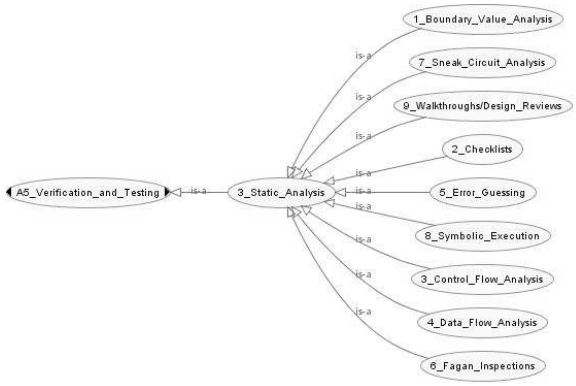

Fig. 4. Verification and Testing methods ontology

These tools are represented on the A-Box level of the ontology, they are instances of the concepts defined using the T-Box in the previous section. A sample tool classification using the ontology description is depicted in Fig. 5.

$$Formal\_Proof(SAL), Formal\_Proof(SPIN),$$
$$Symbolic\_Execution(PolySpace)$$

Fig. 5. Verification and Testing tools classification

IV. Modeling the development process

The (domain-specific) development process is formalised using a process model which describes the tasks, input and output artifacts, the roles and tools involved in the development process.

The OMG's Software Process Engineering Metamodel (SPEM) specification defines a formal representation of business processes (including development processes). The Eclipse Process Framework [2] supports this specification and is proposed in our environment to model the processes.

Using this framework the process designer

- can construct the specific development process, and can assign the available tools to the tasks of the process,
- can choose from the available toolchain patterns.

The tasks of the process implement particular methods that are classified using the ontology that describes the method hierarchy.

In order to support the logical reasoning the process should be represented using ontology. The constructed process model can be transformed into *process ontology* based on the Business Management Ontology (BMO) [4], that supports modeling business processes using mathematical formalisation.

The required model transformation will be implemented using the VIATRA model transformation framework [10] developed by our research group. Using this tool model transformation can be designed and implemented between two models according to their metamodels. The VIATRA framework supports importing models from external source, applying model transformations and exporting models into a specific format.

To use VIATRA the metamodel of the SPEM language and the metamodel of the $\mathcal{SHIQ}$ ontology language will be constructed. After that the „SPEM to ontology" model-transformation (based on these metamodels), an importer for the SPEM models and an exporter for OWL ontology format will be implemented.

There are two alternatives in the construction of the $\mathcal{SHIQ}$ ontology metamodel:

- The Eclipse Ontology Definition Metamodel (EODM) [11] can be used, which is available in the Eclipse Modeling Framework (EMF) [12] and can be imported into VIATRA.
- Constructing a metamodel based on the formal language definition of the $\mathcal{SHIQ}$ language.

A simple example development process is shown in Fig. 6. This development toolchain is compliant to the standard since the combination of Formal Proof (implemented by the SAL model checker) and the Symbolic Execution (which is a Static Analysis method implemented by the PolySpace tool) is a valid combination for SIL3 and SIL4.

The classification of the tools used in this example toolchain are depicted in Fig. 5.

V. Assessment of domain-specific development toolchains

According to the approach described above, all of the tasks, the tools and thus the development processes are characterized using the concepts represented in the ontology.

Using the concepts defined in the ontology, the necessary and sufficient conditions for the selection of methods and the dependency on the safety integrity level should be represented. Two alternative solutions are discovered:

- The conditions can be described using some ontology query language, for example the New RacerPro Query Language (nRQL) [8] or the RDF Query Language [7]. The query should check whether the required combination of the methods are included in the process model. Moreover, the ordering of the methods can also be formulated using these query languages. [15]
- A new concept can be introduced in the T-Box of the ontology. This concept (`CompliantProcess`) represents the set of processes, which satisfy the requirements described in the domain specific standard. So, the requirements can be expressed during the concept definition phase using the $\mathcal{SHIQ}$ ontology description language. The `CompliantProcess` concept will be implemented as a subset of the `Process` concept (which represents the development processes) and restrictions will be defined for it as logical expressions based on the requirements.
  After such categorization a simple A-Box query can be executed in order to find out whether the investigated process is a member of this concept or not.
  The „`(retrieve (nil) (myprocess CompliantProcess))`" example nRQL query answers the following question: Is the process named `myprocess` a member of the concept `CompliantProcess`?

Accordingly, the standard conformance of the selection of methods and their supporting tools in the development process can be checked using an ontology reasoner. In our research we used the Protege [6] ontology modeling tool and the Racer Semantic Web Reasoning System [5]. This way the assessment can be supported by reusing existing formal methods and checker tools.

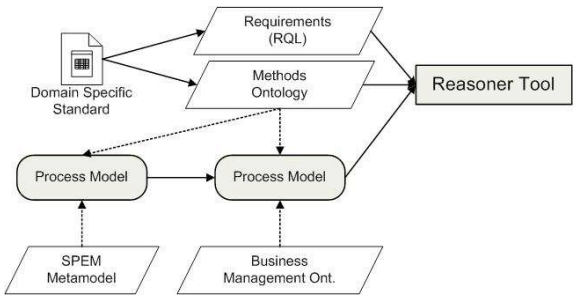

Fig. 7. The assessment process

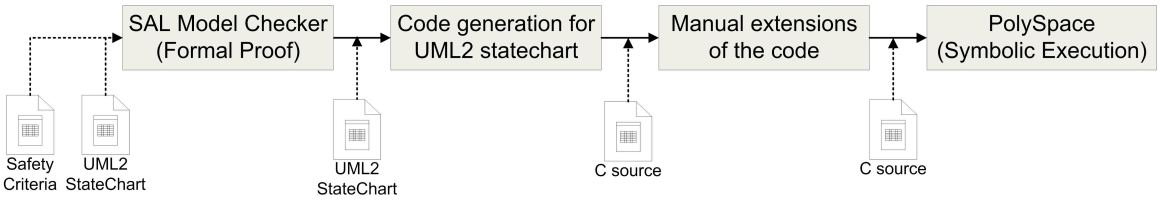

Fig. 6. Sample development process

In Fig. 7 the assessment process is depicted. Based on the domain specific standard the Methods Ontology and the Requirements can be constructed. The development process can be modelled using the SPEM Metamodel and this Process Model should be transformed into a model that is based on the BMO. Finally, the standard conformance of the development process can be checked using the reasoner tool.

VI. Future Work

The used tools will be integrated into an *assessment toolchain* in order to support automatic execution of the steps starting with the SPEM model transformation into ontology based models and finishing with the logical reasoning. A graphical user interface (GUI) will be implemented.

During the standard based assessment of development toolchains not only the used methods or the ordering of the tools are important. The standard specifies that safety arguments are required during the certification process. This safety arguments communicate the relationship between the evidence and objectives.

There is a model based representation of the safety arguments based on the Goal Structuring Notation (GSN) [16]. This notation explicitly represents the individual elements of the safety arguments and the relationships that exist between these elements and the requirements.

The arguments can be ordered into a *hierarchical breakdown structure*. There could be some parts of these arguments that are produced by tools in one step or by toolchains in multiple steps.

The hierarchical breakdown structure can be modeled using a logic description language and the produced arguments can be included in the model of the development process. SPEM supports this by allowing to include artefacts in the development processes. Based on this structure, the reasoning process should be extended in order to check the availability of the expected safety arguments.

At this stage, the proposed assessment process is able to check whether the constructed development process satisfies the requirements described in the domain specific standards. The assessment process will be extended to support the construction of development precesses by identifying missing tools, methods and safety arguments. Based on the process model and the available tools in the repository missing methods could be identified and hints could be given about the supporting tools as well.

VII. Conclusion

Formalisation of requirements is a wide research area. In this work an approach is proposed to formalise the requirements of development processes. This approach forms the basis for the assessment of the standard compliance of specific toolchains and provides support for the process designers to construct certifiable development processes. This work will be used in our MOGENTES project [9], where a tool integration framework is developed.

References

[1] CENELEC EN 50128: *Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems.* URL: http://www.cenelec.eu
[2] *Eclipse Process Framework project*, URL: http://www.eclipse.org/epf/
[3] Sean Bechhofer. OWL web ontology language reference. W3C recommendation, February 2004.
[4] Dieter E. Jenz *Defining a Private Business Process in a Knowledge Base* URL: www.bpiresearch.com
[5] *Racer Semantic Web Reasoning System and Information Repository* - http://www.racer-systems.com/
[6] *Protege ontology editor* http://protege.stanford.edu/
[7] Gregory Karvounarakis and Sofia Alexaki and Vassilis Christophides and Dimitris Plexousakis and Michel Scholl: *RQL: A Declarative Query Language for RDF* - Proceedings of the eleventh international conference on World Wide Web (592-603), 2002
[8] RacerPro Users Guide: The New RacerPro Query Language
[9] MOGENTES project, URL: http://www.mogentes.eu
[10] VIATRA model transformation framework, URL: http://wiki.eclipse.org/VIATRA2
[11] Eclipse Ontology Definition Metamodel, URL: http://wiki.eclipse.org/MDT-EODM
[12] Eclipse Modeling Framework, URL: http://www.eclipse.org/modeling/emf/
[13] SAL model checker - URL: http://sal.csl.sri.com/
[14] SPIN model checker - URL: http://spinroot.com/
[15] Zhijung Zhang: Ontology Query Languages for the Semantic Web: A Performance Evaluation - MsC Thesis URL: http://www.cs.uga.edu/ jam/home/theses/
[16] Tim Kelly and Rob Weaver: *The Goal Structuring Notation - A Safety Argument Notation* - in Proc. Dependable Systems and Networks 2004, Workshop on Assurance Cases, July 2004