7

Lightweight Formal Analysis of Requirements

András Pataricza[1], Imre Kocsis[1], Francesco Brancati[2], Lorenzo Vinerbi[2] and Andrea Bondavalli[3,4]

[1]Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary
[2]Resiltech s.r.l., Pontedera (PI), Italy
[3]Department of Mathematics and Informatics, University of Florence, Florence, Italy
[4]CINI-Consorzio Interuniversitario Nazionale per l'Informatica-University of Florence, Florence, Italy

Requirements are the core work items of the design and checking workflow target safety critical systems. Accordingly, their completeness, compliance with the standards and understandability is a dominant factor in the subsequent steps. Requirements review is a special kind of Independent Software/Systems Verification and Validation (ISVV). The current chapter presents methodologies to use lightweight formal methods supporting experts in a peer review based ISVV.

7.1 Introduction

The quality of requirements dominates the efforts of a design process especially in the case of safety-critical applications (see Chapter 6). As described in the previous chapter in details, the effort and quality of the ISVV heavily depend on the input quality of the work items submitted for review by the customer. Frequently a significant part of the efforts is wasted to basic activities similar to the data cleansing phase in the field of data analysis regarding their level and ratio (which can reach a few tens of percents). For instance, ill-structured documents, inconsequent and non-conformant with the standards

use of terminology all require expert effort to be checked although they do not form the essence of the assessment.

Correspondingly, the exhaustiveness of the checks performed has a major impact on all the activities relying on the completeness, standards compliance and integrity of them. This way, requirement review has to be as thoroughgoing as possible. However, this part of the workflow benefits only to a moderate extent of the advantages of Model-Based System Engineering (MBSE) and formal methods due to the typically conservative (informal) or at most semi-structured text based formulation used mostly in the industry.

The objective of the current chapter is the presentation of an approach targeting a gradual introduction of MBSE and formal methods to requirement checking. The introduction of easy-to-use methods simultaneously assures an increased productivity and quality without the need of a single step introduction of a complete framework or specialized skills in formal methods.

The chapter introduces the basic modeling concepts as defined by standards. The subsequent section presents techniques carrying out extended syntactic analysis over the requirement documents. After addressing change management in iterative requirement design/modification-checking workflows the closing section deals with the integration of the measures described into the ISVV.

7.2 Objective

Our objective is supporting dominantly peer review based ISVV executed by SMEs. Typically, highly-qualified experts constitute the personal of such companies. Reviewers usually are very familiar and knowledgeable of the application domain without deep skills in advanced formal methods.

Accordingly, our evolutionary approach is less ambitious, than a revolutionary one exploiting the full potential of mathematical proof of correctness methods. It follows the paradigm of hidden formal methods in which the user of the tool gets the support from a built-in intelligence, but the working environment has no or very moderate changes compared to the traditional one.

Our approach does assume either an end-to-end MBSE or a complete automation of the workflow; however, it can contribute to a significant effort saving by reducing the overhead originating in document cleaning, managing the progress of the assessment including checking its completeness.

This way, the efforts of the experts can be focused on the hardcore problems related to technical evaluation deliberated of the majority of the pure mechanical tasks not requiring their expertise.

The subsequent sections address the three major questions in improving ISVV:

- How to create interchangeable and well-structured documents out of traditional unstructured ones?
- How to create a domain-specific working environment out of a traditional one by adding quality improvement and checking measures based on hidden formal methods?
- How to improve the convergence of requirements by change management in iterative design-ISVV workflows?

7.3 ReqIF and Modeling

The requirements play an important role in cooperating between final product manufacturers and part suppliers, as these provide the basis of outsourcing and acceptance tests between them. This way, the exchange of requirement between cooperating partners is a crucial part for instance in the automotive industry demanding relatively low interaction times.

The Object Management Group (OMG) developed an open standard [1] called Requirements Interchange Format (ReqIF) to assure interoperability between cooperating partners (see Figure 7.1). This standard is open toward different design and checking technologies thus it is a natural candidate to information exchange between designer and independent software/system

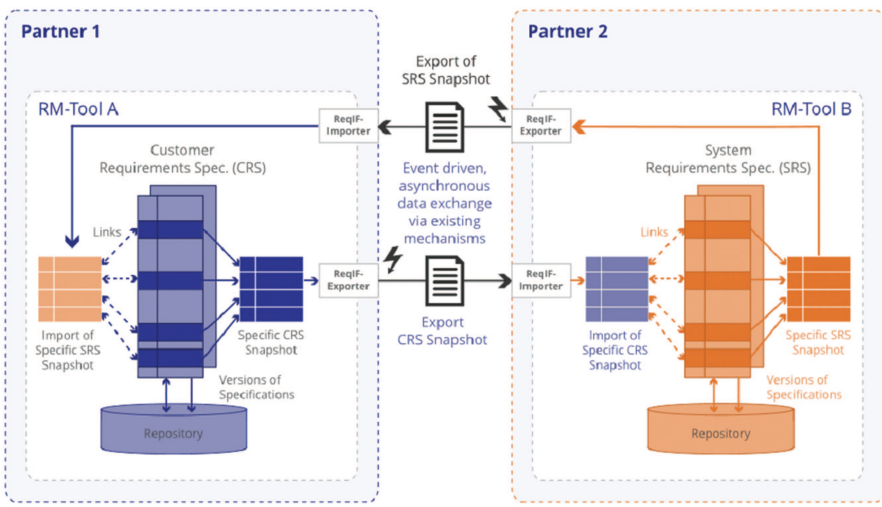

Figure 7.1 ReqIF based information exchange.

verification and validation (ISVV). OMG ReqIF provides a well-regulated set of rules and protocols for cooperation.

Requirements Interchange Format has wide support regarding open and commercial requirement design and management tools. Also, leading vendors put requirements as the core entity of the entire design and checking workflow. Advanced MBSE based frameworks integrate requirement management with design and test. Traceability is a priority concept in ReqIF.

The following summary presents the main benefits of ReqIF as an exchange model language for ISVV based on the top-level constructs in its meta-model.

Requirements Interchange Format supports the exchange of core content labeled by a header sufficiently detailed to identify the document itself and optionally tool specific extensions from other information sources, like results of evaluation (Figure 7.2).

The specification is the core content of a ReqIF instance associated with specification types, objects and the relations between them (Figure 7.3). The notion of links (called here "SpecRelations") assures the traceability of the requirement model to other artifacts. The metamodel supports hierarchical composition of requirement sets, like a well-structured description of safety cases derived by specialization from a standard, e.g., as an evidence list and interlinked supportive arguments.

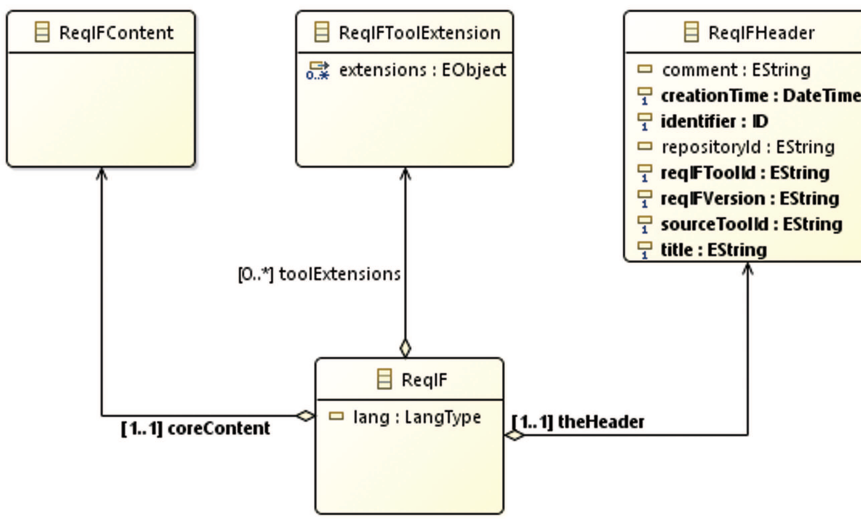

Figure 7.2 Exchange document structure.

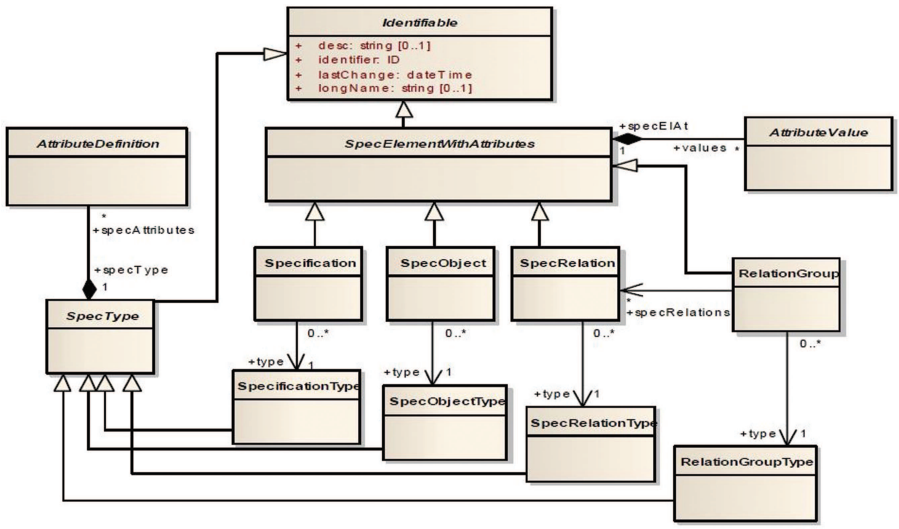

Figure 7.3 Specifications, requirements, and attributes.

The most popular infrastructure for ReqIF is Eclipse RMF (Requirement Management Framework) [2]. ProR [3] an open-source editor to edit structured requirement documents.

Definition of data types, including enumeration types, supports the creation of a domain-specific MBSE-styled model representation.

For instance, the different Safety Integrity Levels form a core enumeration datatype of the form of $\{SIL0 \ldots SIL4\}$. Traditional type checking assures the avoidance of omissions or ill-specified values in the corresponding field.

Syntax-driven editors constrain the designer to use only such values in the field, that comply with the datatype definition.

However, turning a column in an Excel worksheet from the general string type to one out of the predefined values forming the enumerated data type implements simply the same principle. Such constraints simply prohibit entering a wrong value into the instance model corresponding to the application under development.

Moreover, this tiny example indicates a further opportunity in separating the duties: If a domain and modeling specialist designs the Excel template, he could embed the terminology, structure, etc., from the standards, as well. The application designer filling out the template with the content corresponding to the particular application under development will face his traditional working

environment with the hidden type model and check already embedded by the expert.

At the same time, the example pinpoints the limitations of a pure ReqIF-based working environment design approach, as well. Implementation of complex relations necessitates low-level (e.g., VisualBasic) programming and it benefits of modeling only by starting from a proper blueprint, which implies all the drawbacks of traditional programming.

Bidirectional communication between cooperating partners was a primary design objective of the OMG ReqIF standard. In the context of ISVV, this offers the opportunity of using it in the ISVV-to-developer communication for feeding back the review results in an entirely standards compliant way. This way, the iterative process can benefit from the rich navigation and traceability supporting features of ReqIf.

7.3.1 Domain Conceptualization

The industrial success of ReqIF in the inter-party communication in product design makes it a natural candidate in developer-to-assessor cooperation and model-based ISVV, as well. Moreover, as ReqIF documents carry both the instance model and its respective metamodel, they can harmonize of requirement design and ISVV.

At the top end, ReqIF-based requirement modeling serves as the starting point of sophisticated methodologies aiming at correctness by design (like RODIN – Rigorous Open Development Environment for Complex Systems [12] transforming specifications into formal Event-B models). However, the introduction of heavyweight formal methods into ISVV, a single phase of the product development process faces serious obstacles regarding skills, and in the overwhelming majority of ISVV tasks, it has an improper modeling effort/benefit ratio.

Our approach uses ReqIF similarly for information exchange, as this assures a well-structured requirement set. Lightweight modeling should complement the methodology of customization of the ReqIF metamodel and work environment of the requirement composer to a particular product or product family using MBSE. Finally, the customized work environment accommodates traditional, manual, design, and V&V methods, as well.

Ontologies serve as primary candidates for semantics based unification and conceptually clean metamodel design [4]. Ontologies are formalized

vocabularies of terms covering a specific domain. They define the meaning of terms by describing their relationships with other terms in the ontology. They classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on their use by providing formal naming and definition of the types, properties, and interrelationships [5].

Knowledge organization, complexity reduction, and problem solution all use ontologies for a variety of fields ranging from the Semantic Web, through systems and software engineering to such non-technical fields, as library science. The main use case of ontologies is conceptual data integration.

The driving force behind their standardization of formats (RDF and RDF Schemas, OWL) is the World Wide Web Consortium (W3C). The formats support interoperability, information fusion, and interchange.

MBSE largely depends on metamodeling (UML and derivatives). Metamodeling and ontologies are two different, but mutually transformable approaches[1] to modeling language and model construction. Both paradigms focus on the description of the relations between concepts, checking of the compliance of instances (individual models) with their respective parent metamodel or upper ontology.

In contrary of usual metamodels, ontologies have a precise semantics regarding mathematical logic, for instance in ISO/IEC Common Logic [6]. Ontology tools have built-in functions checking the completeness and consistency of the models, and the correspondence of subontologies (specializations) and instances to their upper ontology (subsumption check).

The gradual introduction of hierarchical and relational elements into the model following a vocabulary–taxonomy–ontology process results in an ontology corresponding to a particular standard. Such an ISVV ontology consolidates notions and their mutual relations defined in standards as concepts.

Ontology processing has supportive mechanisms for information fusion by virtually merging multiple, physically separate ontologies. Starting from multiple ontologies representing different viewpoints facilitates aspect-oriented modeling.

---

[1] Theoretically, not all ontologies have an explicit metamodel counterpart, but the subclass of ontologies referred in the current chapter is subject of metamodeling based design. For instance, the Object Management Group (OMG) offers a bridge in the form of an "*Ontology Definition Metamodel*" [13].

The requirement set related to a particular application (legacy documentation, source code, and comments, etc.) are then instances of this ontology. This way the interdependence of entities and V&V steps managing them is explicit.

7.3.2 Integration with Existing Practice of ISVV

A (slightly simplified and obfuscated) real-life example taken from a railway hazard analysis project serves as motivating example.

The railway is a safety-critical domain; various safety measures designed into the system address the hazards that pose an unacceptable risk; these have to be proven to mitigate the various risks to an acceptable level.

The assignment of so-called Safety Integrity Levels ranging from 0 to 4 classify safety instrumented systems and functions. Each level has an associated interval of probability of failure on demand of the safety function, what translates to an overall risk reduction capability.

Designers of the original documentation used a plain, unstructured list of hazards as the input for risk analysis (Figure 7.4). However, structuring the potential causes indicates clearly its flaws. At first, the introduction of the abstract concepts "*Subject*" and "*Impact*" separates the different aspects related to a hazard event (Figure 7.4).

Aspect weaving in the form of interrelating them derives the individual categories, like "*Line Controller Death*." The inclusion of "*No Hazard Event*" and "*Fire*" do not fit into the scheme. The list of hazards is still incomplete w.r.t cardinality constraints. For instance, the standard may require the complete coverage of all potential hazard events by evaluating all "*Subject*" and "*Impact*" combinations. Automated reasoning reveals that the "*Staff OnBoard Death*" category lacks the considerations.

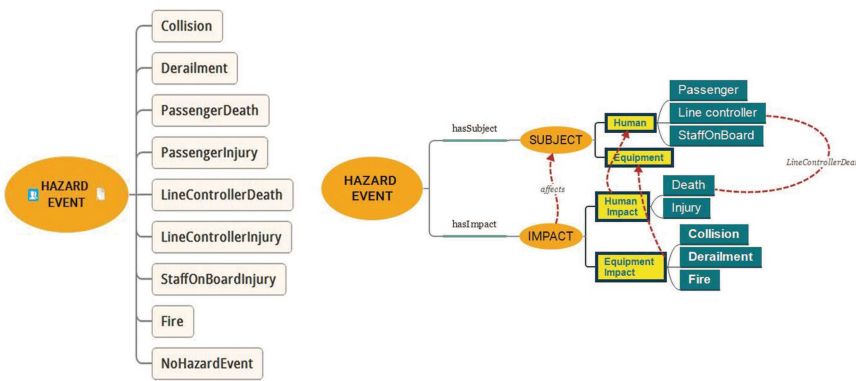

Figure 7.4 Unstructured and structured model.

At the same time, this model is easy to maintain. For instance, after the introduction of the notion of a "*Driver*" as a separate category, inherence mechanisms can derive the two subcases "*Driver Death*" and "*Driver Injury*" without touching other parts of the model.

Moreover, the design and ISVV workflows may rely on external information sources, as well. Information fusion necessitates the unification of the concepts of the different data sources by establishing the correspondence between their notions.

For instance, risk analysis should cover all the hazards above a given frequency of occurrence, which necessitates the inclusion of historical statistical data from external data sources (like [14] in Figure 7.5). Their integration into the ontology can follow the same unification approach, as in [7] based on mapping the notions in different models after some elementary operation (like calculating totals when aggregating overly fine granular statistical data).

Note, that the process of information fusion is an important engineering task and not a pure semantic matching of two models. Apparently, resolution of the two models differ merging different categories in the statistics into a single concept in the model of hazard events assumes a similarity in their occurrence and impacts. Aggregation of categories is at the same time an input specification for the underlying summation of frequencies of their occurrence.

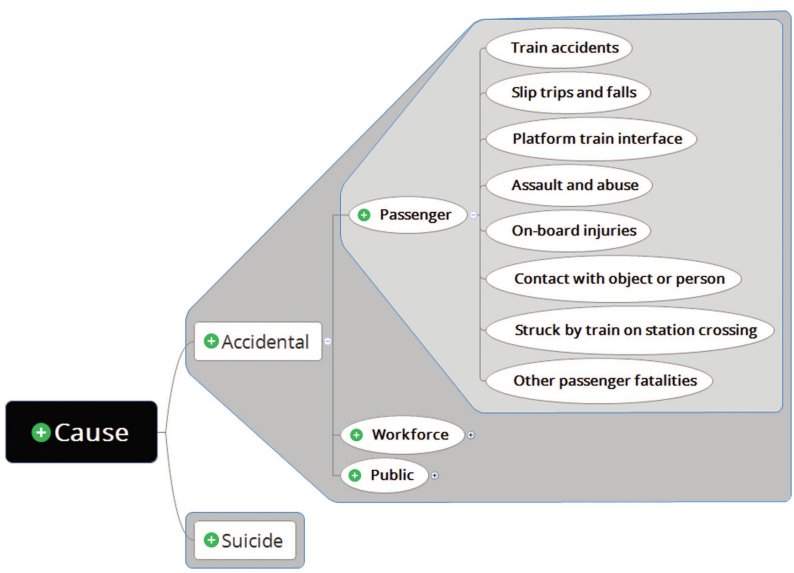

Figure 7.5 Causality statistics structure.

Such an interrelation of statistical data and the input model of hazard analysis support augmentative maintenance of the model. The appearance of a new category in the statistics (e.g., security) with no counterpart in the hazard event ontology pinpoints that the later one is not up-to-date.

7.4 Requirement Change Propagation

Our motivational example comes from the railway domain loosely based on an actual change scenario, similarly as the example above. It highlights the importance of lightweight formal methods from a further aspect, change management. For didactical as well as legal reasons, the case presented here is very heavily simplified and sanitized from multiple aspects.

The SIL of a function has a fundamental impact on its development cost and time, as higher levels require increasingly sophisticated V&V activities. Consequently, during requirement change impact analysis it is essential to correctly identify whether a requirement change indirectly causes SIL changes in a specification through change propagation.

7.4.1 Original Specification

Our example demonstrates how the changes in the requirement set of a Central Traffic Control system have a propagation effect in the whole specification.

Keeping station area traffic safe is a complex problem involving many tracks and switches in a complex manner, and the risks stemming from a significant number of hazardous situations have to be mitigated. Trackside signals regulate station area traffic allowing or denying entry to a track or (switching) point. Classically, the control of the traffic through the signals has been performed by local personnel and systems. The main means of risk mitigation is *signal interlocking*: a separate system overrides any traffic control command that would lead to a hazardous signal configuration. (For instance, giving a "clear" signal at the same time at two entry points of an interlocking.) Signal interlocking can be overridden in the local traffic control system under strict operational rules, e.g., a switch with broken switch state monitoring correctly halts traffic; however, to resume traffic, local personnel has the situational awareness and authority of a temporal override of the associated signal interlocking.

Traffic control has been and is being centralized worldwide To increase operational efficiency, a Central Traffic Control (CTC) system manages the

traffic at multiple stations. CTC can be an overlay to the existing systems without substituting the local traffic control and the remote CTC "pushes its buttons" instead of local personnel. Local signal interlocking is left unchanged, too.

The "original" specification on Figure 7.6 represents a simplified excerpt from the specification of such a system. Importantly, the CTC does not have the full authority of local personnel; it is not allowed to issue signal

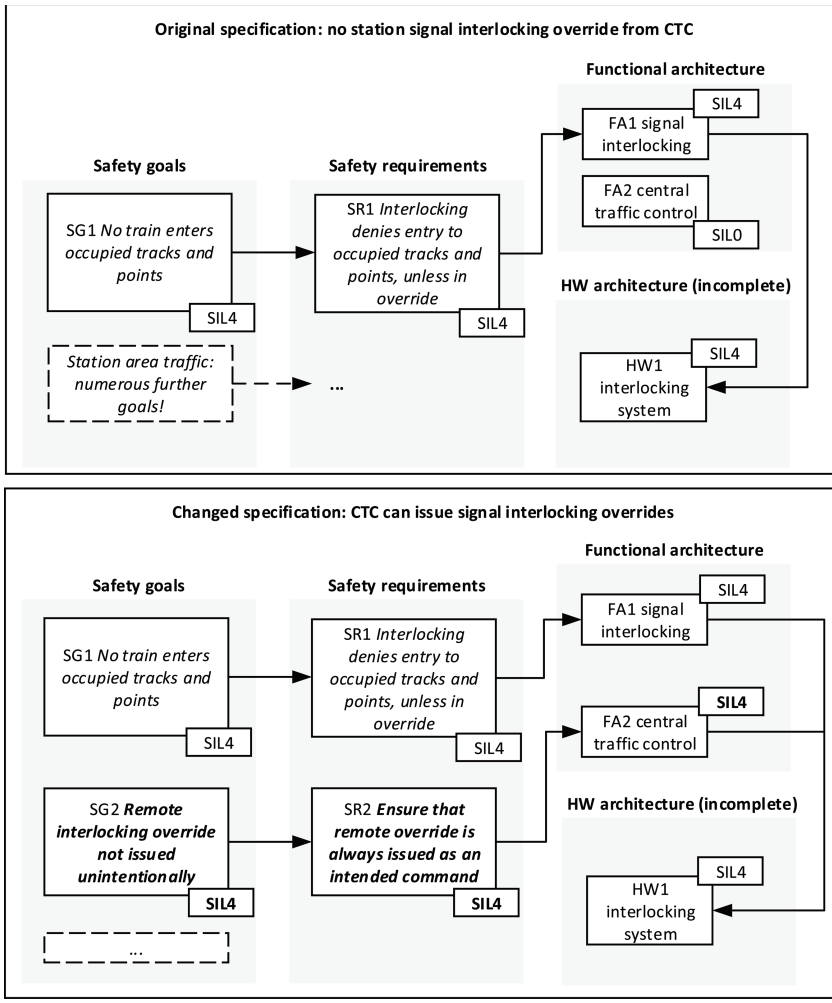

Figure 7.6 The original and changed specification in our example.

interlocking override (more generally, safety-critical) commands. As a result, the process of setting up safety goals, decomposing them into safety requirements and mapping those onto elements of the functional architecture identifies that it is SIL0 (not safety critical). On the other hand, notice how the high-risk mitigation capability requirement (SIL4) is carried over from the safety goal to the interlocking system.

7.4.2 Changed Specification

This specification has the chance to lead to a highly safe system that conforms to the legal requirements on safety. However, an operator is also concerned about operational efficiency. Let us assume that the operator finds the above specification too restrictive; in many circumstances, override situations can be managed acceptably safely, even if the override command is issued remotely. However, some characteristic hazards have to be avoided [8]; one of them is the CTC issuing override commands unintentionally. (The CTC is usually a complex, software-based system, where operators manage multiple stations of a geographical region with reduced situational awareness due to their remote location.) This leads to the specification excerpt depicted in Figure 7.6 as the changed specification.

The key difference between the two specifications from change impact analysis is that the central traffic control became a safety-critical component. Risk mitigation assumes the absence of override commands issued by the central traffic control unintentionally.[2] This change in SIL has further propagating effects; the V&V activities associated with the architecture, interfaces and implementing components of central traffic control have to be revisited.

7.4.3 The Change Impact Propagation Method

Requirement engineers have to evaluate the propagating effect of changes and rework the specification accordingly. This task involves two major phases.

- ***Suspicion marking through change impact propagation***. The directed dependency graph of the specification is traversed starting from the initial changes introduced into the specification. Dependencies and requirements that may have to be changed as an effect of the original change are marked as *SUSPICIOUS*. After that, specification objects

[2]There are many ways to ensure this, regarding the operators as well as the software/hardware system; discussing these is not in the scope of this chapter.

that are connected to *SUSPICIOUS* ones are evaluated and potentially marked, too in a transitive manner. Effectively, the *SUSPICIOUS* marking is "propagated" in the reachability subgraph of the originally changed elements. The resulting *change impact cover* – the subgraph defined by the vertices marked *SUSPICIOUS* – is passed on to marking processing.

- ***Processing marking***. One by one, the suspicion-marking of the marked dependencies and requirements has to be either accepted or refuted. If accepted, the appropriate specification change has to be designed and performed.

We are mainly concerned here with the first phase, although the value-based change impact propagation we introduce gives guidance to the second one, too.

- In practice, manually performing the first activity is a repetitive, time-consuming and error-prone task even for moderate size specifications.
- The best of breed modern requirement management tools support topology-based propagation: anything that is connected to a specification element marked *SUSPICOUS* is *SUSPICIOUS*, too.
- Some modern tools begin to support type-based propagation. In this case, marking is propagated only along the configured types of dependencies and only upon the configured types of requirement attributes becoming *SUSPICIOUS*.

Type-based propagation is a powerful tool to reduce the extent of the change impact cover in the specification. Observe that on Figure 7.6, textual description change along the <*SG2*, *SR2*, *FA2*, *HW1*> trace does not propagate into the functional architecture due to the safety requirement mapping nature of the (*SR2*, *FA2*) link. On the other hand, SIL change does propagate, as *FA2* got connected to a new safety requirement; thus, its SIL has to be potentially (and in this case, also actually) modified.

Value-based propagation can further reduce the extent of the change impact cover. In addition to types, it also takes into account the nature of the propagating changes as well as the current values captured in each requirement. Notice that HW1 has not to be changed, although FA2 has been mapped to it. The reason is that although it got newly connected to a (newly) high-SIL function, it already has the highest SIL level. Thus, during marking, propagation can safely stop here. Figure 7.7 demonstrates the relationship between the change impact cover extents and the resolution of propagation.

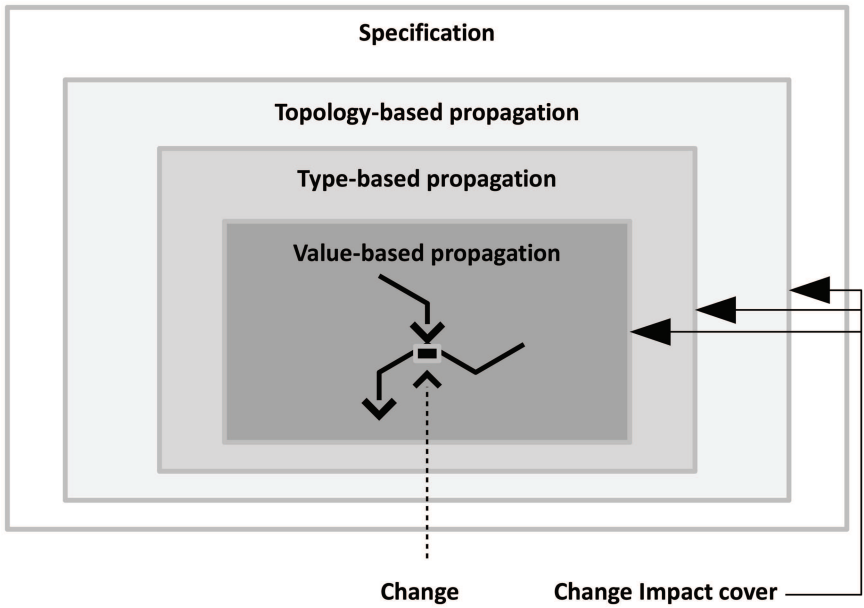

Figure 7.7 Propagation resolution and computed change impact cover extent.

Independently of the category, we have to note that propagation does not necessarily happen only "forward" or "downstream". The change of a requirement may impact its parent (containment-wise), not just its children; and it may impact the sources of its incoming traceability links, not just the targets of its outgoing ones. Tooling supports the user to configure the directionality of propagation; this is a largely orthogonal concern to the propagation resolution. For the sake of simplicity, in the following, we focus on the forward direction unless otherwise noted.

7.5 Abstraction Levels of Impact Propagation

We have argued informally that there are three major categories of change impact propagation from resolution. In this section, we describe and compare these categories using a simple example.

Let us consider the rich requirement structure in Figure 7.8. In addition to an SIL attribute, our requirements can have a priority attribute, too. In the context of this example, priority expresses the importance of overall operational efficiency with the levels `HIGH`, `MEDIUM` and `LOW`. It aggregates

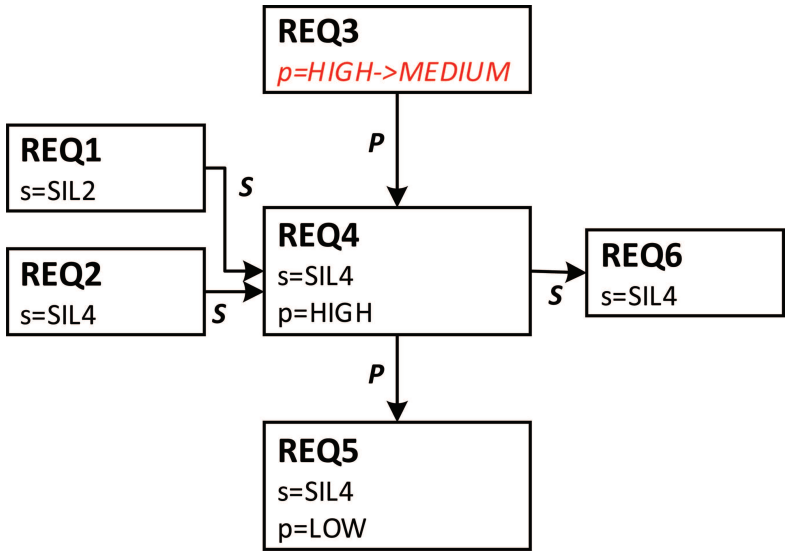

Figure 7.8 Example rich requirement structure for propagation categorization.

some concepts, including maintainability, time to repair and cost of operation. This priority concept is largely independent of SIL, and while the safety level is an absolute requirement, priorities are subject to business considerations and not critical to meet.

The example requirement model uses

- containment (parent relationship, denoted by P),
- SIL-mapping traceability links (denoted by S) and
- SIL and priority level attributes for the requirements.

For the purposes our example, we assume that the following consistency rules are applied during requirement management.

- **Rule 1**. Any requirement that has an incoming "safety mapping" (S) traceability link has an SIL attribute, and its value is the maximum of the SIL values at the source requirements. For codification, the requirement engineer should be able to derive this rule from the process definition and the safety standards that have to be applied.
- **Rule 2**. A prioritized requirement must have only prioritized descendants. This value can be only less or equal than that of the parent. For codification, the requirement engineer should be able to formulate this rule as a locally used and observed rule.

Let us emphasize that these rules are for demonstration purposes. SIL value constraints along traceability links can be much more complicated in the general case. The handling of priorities also represents only one possible choice; among others, even its exact reverse may be justified in a specific project. Our modeling approach and the subsequently introduced solution method can support almost arbitrary rules. We also handle the potential non-determinism of the rules.

The change we will be concerned with is modifying the priority of `REQ3` from `HIGH` to `MEDIUM`. Figure 7.9 demonstrates propagation for the three categories.

7.5.1 Topology-Based Propagation

We can propagate the change along the outgoing dependencies (containment and traceability links) of the requirement, marking requirements transitively as *SUSPICIOUS*. This approach is commonly referred to as *topology-based* change propagation. In addition to attribute changes, the creation of new dependencies as well as deletion of existing ones (through deleting the source/target requirement or otherwise) is seamlessly supported.

7.5.2 Type-Based Propagation

The next level is *type-based* propagation. Dependencies have types, as well as the attribute of the originally changed requirement that is changed. We can filter propagation for dependency type as well as changed attribute type. We reflect the changes that are allowed to propagate into the first-level dependents by marking the dependents or some of their own attributes as *SUSPICIOUS*. We can then perform propagation from this first level transitively by propagating the requirement or attribute *SUSPICIOUS* marking using the same configurable filtering and configurable attribute marking mapping mechanism. In the context of the example of Figure 7.8: for priority changes, we propagate the *SUSPICIOUS* marking only along the descendants of the changed requirement. Priority attribute markings are mapped into priority attribute markings, as there is no logical dependency between the two attributes in this case. Note that in addition to the orthogonal analysis of attributes, this as well as the next category supports conjoint analysis – we can express when the change of an attribute propagates to another.

Dependency changes are handled similarly to topology-based propagation.

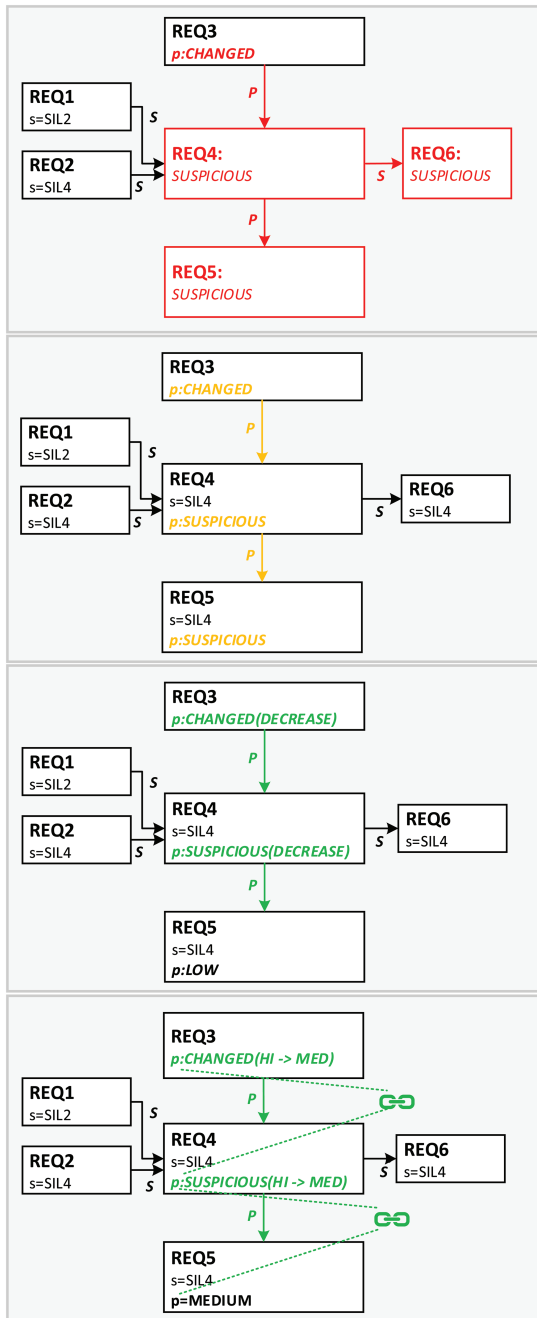

Figure 7.9 Change impact propagation categories.

7.5.3 Value-Based Propagation

Type-based propagation is a powerful tool for controlling the extent of change propagation in specifications that have a dense dependency structure; however, it still does not take into account the *kind* of the change in the value domain. Change can be described either qualitatively or in specific terms; a qualitative description for priorities would be e.g. declaring its *increase* or *decrease*, while the specific description is either the new value or the old-new value pair. We establish the category of *value-based* change impact propagation with these two subcategories.

In the qualitative case, observe that if Rule 2 is known, it means that the priority *increase* change of a parent does not *need SUSPICIOUS*-marking on the children. However, when it *decreases*, priorities in the children *may* have to be revisited. We can say even more: when the priority at the next level is `HIGH`, we are certain that changes are necessary; conversely when it's `LOW`, we are certain that propagation stops here. A somewhat similar qualitative logic exists for the SIL mapping in our case; at level 4, any increase upstream will not have any impact, while a decrease may require requirement reconsideration.

The downside is that the rule set that is to be used has to be defined; however, with predefined templates, this promises to be a manageable overhead.

The next logical step is to consider propagating the specific change and computing the specific local changes that may be necessary to be made. We call this approach *specific value based change propagation*. On paper, this idea seems not too far-fetched (see the example in Figure 7.9). However, it requires formulating explicit, value-specific consistency rules. More specifically, propagation needs value *change* consistency rules that connect allowed changes in localized requirement contexts (in the simplest, the allowed attribute co-changes at the two ends of a typed link). However, it is easy to see that the most of such change consistency rules can be transformed into value-specific specification consistency rules and vice versa. In this last case, the line between propagation and marking processing becomes very blurred, as essentially specific change candidates are computed as a marking during propagation.

We treat this last category as a theoretically interesting option; however, it is one that has little immediate value to the practice in an industry that doesn't even use type-based change propagation in a widespread way yet.

We have conducted an analysis of a sample of the best-of-breed requirement management solutions, to determine the extent and sophistication to which they support assessing the propagation of requirement change impacts. Topology-based propagation seems becoming available. Type-based propagation is still a novel feature, available, e.g., in Rational DOORS Next Generation. Value based propagation (qualitative or otherwise) is practically non-existent yet.

7.6 Resolution Modeling with CSP

To establish a common, computable framework for the first three categories above, we define them declaratively as finite-domain Constraint Satisfaction Problems (CSPs) [9]. The motivation is that many sensitivity analysis tasks in error propagation assessment and test generation are known to be definable and also solvable this way – and tracking the propagating effect of requirement changes is very similar to tracking the potential effects of faults in a system.

In CSPs, a finite set of variables, each with a nonempty domain, is subjected to a set of constraints. Each constraint is a relation that specifies the permissible value combinations for a subset of the variables; a solution of the CSP is such a value-assignment of the variables that satisfies each constraint. An important category of such problems is finite-domain CSP, or csp(FD); in this case, the variables are discrete and have finite domains. This way, csp(FD) expresses combinatorial search style problems.

The power of csp(FD) is that it can be used to declaratively specify a problem and letting one of the mature, optimized and very sophisticated existing tools to look for a solution (or enumerate all solutions). Tools widely recognize a standard set of composable "simple" constraints (linear arithmetic equalities and inequalities, Boolean arithmetic, etc. over the declared variables) and so-called global constraints, too. The latter involve a potentially large number of variables and need specific algorithmic optimization (for an exhaustive list, see the Global Constraint Catalog [10]). Constraint problems have a widely recognized standard representational language in the form of XCSP3 [11].

Change impact propagation problems can be easily represented as a CSP. We declare the marking of each requirement, attribute and dependency (containment and traceability links) as a variable; define the possible marking value set for each; describe propagation as constraints and lastly introduce the constraints for the performed changes. Table 7.1 gives an outline of this process.

Table 7.1 Comparison of change impact propagation categories

		Constraints for the Dependencies		
Cat	Requirement/ Dependency Marking Literals	If Dependency *d* with Type *t* is ...	...then	Further Constraints
1.	Reqs: *CHANGED, SUSPICIOUS, INTACT*	Not *DELETED* or *ADDED*	Target of *d* not *INTACT*	Only the actual changes can be *CHANGED, DELETED, ADDED*
	Attributes: N/A	the source of *d* is not *INTACT*	*d* is *SUSPICION_LINK*	Maximize the number of *INTACT* markings
	Dependencies: *DELETED, ADDED, SUSPICION_LINK, INTACT*	Not *INTACT*	Target of *d* is not *INTACT*	
2.	Reqs: N/A	not *DELETED* or *ADDED*	Attributes declared as *a*-propagation target for *t* at the target of *d* not *INTACT*	Only the actual changes can be *CHANGED, DELETED, ADDED*
	Attributes: *CHANGED, INTACT, SUSPICIOUS*	attribute *a* at the source not *INTACT*	*d* is *SUSPICION_LINK*	Maximize the number of *INTACT* markings
	Dependencies: *DELETED, ADDED, SUSPICION_LINK, INTACT*	*a* declared propagation source for *t*		
		DELETED or *ADDED*	Attributes at the target declared for *t* creation or deletion propagation not *INTACT*	
3a.	Reqs: N/A	Rule set that for each dependency type *t*, encodes the relation expressing the together permissible (or ruled out)		Only the actual changes can be *CHANGED, DELETED, ADDED*

Table 7.1 Continued

For each attribute: *NOCHANGE, SUSPICIOUS_CHTYPE1, SUSPICIOUS_CHTYPE2, ...*	• source-side attribute marking values, • target-side attribute marking values, • dependency markings.
Dependencies: *DELETED, ADDED, SUSPICION_LINK, INTACT*	Rules allowed also to incorporate current attribute values.

A few things have to be noted on this framework. For topology- and type-based propagation, the table describes only rules for forward propagation; however, backward propagation rules can be introduced similarly. Notice that type-based propagation introduces attribute marking and discards requirement marking; the latter can be incorporated (with some complexity increase) or emulated by declaring all attributes suspicious.

For qualitative value-based propagation, due to the variability of the rules (that is the intended goal), we can't characterize propagation rules in the same manner. Still, all practically important propagation intentions can be formulated using standard CSP expressions. For instance, propagation of priority increase suspicion for requirement `R1` and `R2` interconnected through a link type `t` can be expressed e.g. as `t_connected (`**`R1_p_marking, R2_p_marking`**`) AND` **`R1_p_marking`** `==` *`SUSPICIOUS_INCREASE`* `AND` **`R2_p_current`** $<$ *`HIGH`* $\rightarrow$ **`R2_p_marking`** `==` *`SUSPICIOUS_INCREASE`*.[3] That said, our ongoing work addresses creating a domain-specific language that simplifies the creation of the sets of rules.

7.7 Conclusions

The OMG Requirements Interchange Format (ReqIF) assures interoperability between cooperating partners. This standard is a natural candidate to information exchange between designer and independent software/system verification and validation (ISVV).

Using ReqIF facilitates (which is occasionally only the question of asking the developers using top-end requirement design tools for providing

[3] Variables are typeset bold, value-literals are typeset italic and `t_connected` is a constraint that we defined based on the specification.

the native ReqIF model in addition to the derived documentation not containing the model) an immediate entry to the benefits of lightweight formal models.

At the same time, OMG ReqIF provides a well-regulated set of rules for the developer-ISVV interoperation and the communication of the assessment results.

Traceability is a priority concept in ReqIF. The option of defining the structure the document, introducing types and well-formedness constraints are all major means to introduce the main concepts of domain-specific MBSE.

Similarly to development, where advanced tools can generate traditional office-like documentation out of their internal ReqIFmodels, ISVV can highly benefit from using RequIF as the core model for communicating ISVV results.

Ontologies provide an easy way to overcome the limitations of ReqIF regarding conceptual modeling. Ontology-based metamodel design is a modern model development paradigm, as its standardized language and development tools implement all the main concepts of complexity management, like the composition of complex ontologies out of simpler ones, hierarchical modeling and aspect weaving. At the same time, their well-defined semantics allows using reasoners.

The simple mathematical background of ontologies, set theory results in a low entry threshold related to skills. The built-in logic reasoners can check the contradiction freedom of a requirement set (by a satisfiability test), and its well-formedness (by a subsumption check), thus deliberating the ISVV of tedious manual checks.

Ontologies are highly standardized. Model formats assure interoperability; moreover, standard transformations exist to the world of metamodeling. As ontologies provide an abstract representation of knowledge, automated export and import tools exist between ontologies and knowledge storage tools like structured semi-formal representations (Excel), relational, object-oriented and graph databases.

Classically, requirement changes involve a significant effort and quality cost, especially if the tooling provides no proper guidance for the reassessment. Intelligent change impact analysis helps properly focusing the assessment after a change by evaluating the propagating effects of the introduced changes. In a properly structured requirement specification with a rich traceability structure, algorithmic analysis can significantly reduce the extent of the change impact propagation cover that analysts have to check.

References

[1] Object Management Group. (2017). *Requirement Interchange Format (ReqIF)*. Available at: http://www.omg.org/spec/ReqIF/ (accessed on 1 March 2017).

[2] *Requirements Management for Eclipse*. Available at: https://eclipse.org/rmf/ (accessed on 1 March 2017).

[3] Eclipse. (2017). *ProR Requirements Engineering Platform*. Available at: http://www.eclipse.org/rmf/pror/ (accessed on 1 March 2017).

[4] Knublauch, H. (2004). "Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL," in *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)* (New York, NY: IEEE), pp. 381–401.

[5] W3C. (2009). *W3C: OWL 2 Web Ontology Language Document Overview*. Available at: https://www.w3.org/2009/pdf/REC-owl2-overview-20091027.pdf (accessed on 1 March 2017).

[6] ISO. (2007). *ISO/IEC 24707:2007: Information technology – Common Logic (CL): a framework for a family of logic-based languages*.

[7] Pataricza, A., Gönczy, L., Kövi, A., and Szatmári Z. (2011). "A Methodology for Stand-ards-Driven Metamodel Fusion," in *Model and Data Engineering: First International Conference, MEDI 2011* (Berlin: Springer), 270–277, Óbidos, Portugal, September 28–30, 2011. Eds L. Bel-latreche and F. Mota Pinto.

[8] Tarnai, G., and Sághi, B. (2006). "Hazard and Risk Analysis of Human-Machine Interfaces of Railway Interlocking Systems," in *7th World Congress on Railway Research*, Montral, Canada, 4–8 June.

[9] Brailsford, S. C., Potts, C. N., and Smith, B. M. (1999). Constraint satisfaction problems: algorithms and applications. *Eur. J. Operat. Res*. 119.3, 557–581.

[10] Beldiceanu, N., Carlsson, M., and Rampon, J.-X. (2012). "*Global Constraint Catalog, (revision a)*." Available at: http://www.diva-portal.org/smash/record.jsf?pid=diva2:1043063 (accessed on 1 March 2017).

[11] Frédéric, B., Lecoutre, C., and Piette, C. (2016). "*XCSP3 Specifications – Version 3.0*." Available at: http://www.xcsp.org (accessed on 1 March 2017).

[12] RODIN. (2017). *Rigorous Open Development Environment for Complex Systems*. Available at: http://rodin.cs.ncl.ac.uk/ (accessed on 1 March 2017)

[13] Object Management Group. (2017). *Ontology Definition Metamodel (ODM)*. Available at: http://www.omg.org/spec/ODM/ (accessed on 1 March 2017).

[14] Government of the United Kingdom, Department of Transport. (2017). *Rail Accidents and Safety Statistics Tables*. Available at: https://www.gov.uk/government/statistical-data-sets/rai05-rail-accidents-and-safety (accessed on 1 March 2017).