Supporting the Validation of Adequacy in Requirements-Based Hazard Mitigations

Bastian Tenbergen(✉), Thorsten Weyer, and Klaus Pohl

paluno – The Ruhr Institute for Software Technology,
University of Duisburg-Essen, Essen, Germany
{bastian.tenbergen,thorsten.weyer,klaus.pohl}@paluno.uni-due.de

Abstract. [**Context and motivation**] In practice, validating functional safety requirements is mainly done by means of reviews, which require large amounts of contextual information about hazards, such as safety goals or the operational conditions under which the hazard occurs. **[Question/problem]** This information is often scattered across a plethora of artifacts produced particularly during requirements engineering and safety assessment. In consequence, there is a risk that not all relevant information is considered during reviews, leading to subjective and misjudged results. **[Principal ideas/results]** In order to improve the consideration of all relevant information necessary to validate functional safety requirements, we propose a diagrammatic representation integrating all relevant contextual information. **[Contribution]** We hypothesize that reviewers are more likely to base their judgment on the relevant contextual information about the hazard, which increases objectivity and confidence in review results. To support this hypothesis, we report preliminary results of an empirical study.

Keywords: Safety requirements · Hazards · Validation · Safety assessment · Mitigation · Adequacy · Safety-critical embedded systems

1 Introduction

During the development of safety-critical embedded systems (hereinafter "systems"), particular emphasis must be placed on ensuring sufficient system safety, i.e. ensuring that during operation, the system's functionality does not lead to harm for human users, external systems, or the environment [1]. During development, safety assessment is concerned with providing objective assurance that all identified hazards are adequately mitigated, i.e. that any operational situation in which the system's functionality leads to harm is sufficiently improbable (cf. [2], [3], [4]). For this purpose, in early phases of safety assessment, initial requirements are subjected to hazard analyses (e.g., Functional Hazard Analysis, FHA [5]) to identify potential hazards and define possible safety goals (see [6], [7], [8]). Safety goals typically describe abstract conditions to be achieved [8], where the concrete implementation is left up to the developer [9]. Before such mitigations can be implemented into the system, it is necessary to refine safety goals into functional safety requirements, which document the conditions and capabilities to mitigate a hazard.

S.A. Fricker and K. Schneider (Eds.): REFSQ 2015, LNCS 9013, pp. 17–32, 2015.
DOI: 10.1007/978-3-319-16101-3_2

As Hatcliff et al. have recently illustrated in [10], "one of the biggest challenges in engineering certifiably safe software-dependent systems is to establish valid [functional safety] requirements," (see [10], p. 189). *Valid* in this sense means that the *right* capabilities and conditions (cf. [11]) have to be specified, which are *adequate* to mitigate the identified hazards. In accordance with [12], we prefer the term *adequacy* over *correctness* [13] in order to honor the non-binary nature of the suitability of requirements for this purpose. Inadequate safety requirements have severe repercussions for the system, as inadequacy may not only result in project delays and extraneous development cost like in non-safety-critical systems [14], but in worst case can result in death [10]. Therefore, Hatcliff et al. argue that "requirements engineering should facilitate the validation needed for [the assurance of system safety]" (see [10], p. 190).

From a requirements engineering perspective, however, therein lies a significant challenge. Validation is in practice often done through reviews, which require large amounts of contextual information about the hazard, i.e. safety goals from hazard analyses and the operational conditions under which the hazard is triggered. This contextual information is not only distributed among the requirements specification, but also across artifacts from safety assessment. Moreover, reviews often depend on the reviewers' understanding of the problem domain [15], [16] and development process [17], [18] as well as the availability and presentation of information to be reviewed [19]. In particular, the widespread use of natural language in development artifacts is seen as detrimental to validation due to inherently poor traceability and the sheer amount of documents to review [19]. Consequently, there is a risk that crucial information to conduct such a review is overlooked, leading to subjectivity and low confidence in review results as well as misjudged adequacy of functional safety requirements. Model-based representations have recently received attention to alleviate these risks by fostering artifact understandability, traceability, and communication about the contained information [20].

In this paper, we seek to support the consideration of the information relevant to review the adequacy of functional safety requirements. We propose a diagrammatic representation which integrates functional safety requirements, hazards, safety goals, and trigger conditions. We hypothesize that by using this integrated representation, more information relevant to the hazard is used during review. In the following, we introduce a running example in Section 2 and fundamental concepts in Section 3. The integrated representation is discussed in Section 4. In Section 5, we present results of a pilot study to support our hypothesis. Section 6 briefly reviews the related work and Section 7 summarizes this paper and gives an outlook on future work.

2 Running Example

We illustrate our modeling approach by means of a simplified Traffic Collision Avoidance System (TCAS) from the avionics industry. The main functionality of a TCAS is to survey the airspace surrounding the own aircraft, to warn the pilots of other aircraft in the vicinity, and to suggest collision threat resolutions. The activity diagram in Fig. 1 shows the functional requirements of the TCAS described in [21].

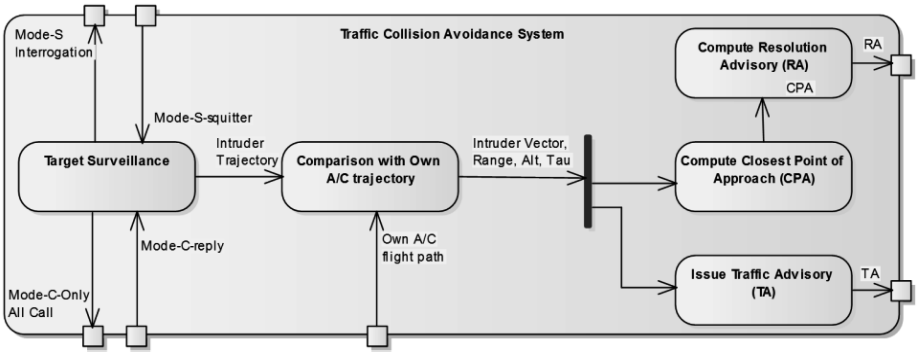

Fig. 1. Functional Requirements of the Traffic Collision Avoidance System

The TCAS consists of five basic functions: *Target Surveillance, Comparison with Own Aircraft (A/C) trajectory, Compute Closest Point of Approximation* (*CPA*), *Compute Resolution Advisory* (*RA*), and *Issue Traffic Advisory* (*TA*). The TCAS may request the own aircraft's transponder to issue a Mode-S-Interrogation, i.e. a signal which requests nearby traffic (such as passenger, cargo, or military aircraft), to reply with uniquely identifying information, course, and altitude (*Mode-S-squitter*). Traffic that is not equipped with Mode-S capability (e.g., some sport planes), will be surveyed using *Mode-C-Only All Call*, which is akin to a regular radar sweep and allows *Target Surveillance* to infer course and altitude from repeated *Mode-C-reply*. Once traffic is detected, the TCAS compares the trajectory of the possible intruder with the own aircraft's flight path and determines the intruder's altitude (*alt*), *range*, directions to the intruder (*intruder vector*), and the time to intercept (*tau*). This information is relayed to the function *Issue Traffic Advisory* (*TA*), which displays the information on a cockpit display, and the function *Compute Closest Point of Approach* (*CPA*). If this function determines that the intruder is a threat to be avoided, the resolution advisory (*RA*) is computed and audio-visually relayed to the pilots. While the TA merely informs the pilots of nearby traffic, the RA advises pilots to climb or descend at once in order to increase separation between the own aircraft and the intruder and prevent a collision. For more details on the functionality of an airworthy TCAS, see [21].

3 Fundamentals

The academic disciplines of requirements engineering and safety engineering[1] are closely related, yet, a consistent terminology between the two has not yet emerged. Therefore, in the following, the basic terms and definitions underlying our approach are introduced.

[1] In the following, we use the term *safety engineering* for the academic discipline, while we use the term *safety assessment* for the activities carried out during development.

One term that is of particular importance, but differs across standards and authors in the field of safety engineering is the term "hazard". In early phases of development, hazards are identified based on functional requirements using key words indicating erroneous behavior (e.g., "fails to operate", "operates inadvertently", "produces wrong output", see [5]). In this paper, we adopt the following definition based on [3]:

Definition 1: Hazard. *A hazard is a set of system states during operation that – together with triggering conditions in the operational context of the system – could lead or contribute to an accident.*

Safety assessment is not only concerned with identifying hazards, but also with ensuring that these hazards are properly mitigated. This means that abstract safety goals conceived during early phase FHA (see Section 1) must be refined into concrete measures to mitigate a hazard. For the purpose of this paper, we will use the more general term "mitigation" and adopt the following definition based on [3]:

Definition 2: Mitigation. *A mitigation consists of a set of functional safety requirements that refine safety goals into concrete implementable measures to avoid a hazard or reduce its harmful effects.*

The term "functional safety requirement" is used in two distinct, but related ways: In some cases, a requirement is often considered safety-critical when it gives rise to a hazard (see, e.g., [2]). In contrast, especially requirements engineering literature often considers safety requirements a type of quality requirement (e.g., [22], [23]), which is in place to achieve a certain level of safety. However, safety can only be achieved when concrete functional safety requirements in the sense of [2] are in place, i.e. concrete conditions and capabilities that, when implemented entirely and without error, mitigate the hazard. To honor this dual role of requirements with regard to safety and to emphasize the functional nature of requirements documenting hazard mitigations, we hence adopt the following definitions inspired by [2]:

Definition 3: Hazard-Inducing Requirement. *A hazard-inducing requirement is a functional safety requirement in the sense of [2], which given triggering conditions in the operational context of the system, cause a hazard.*

Definition 4: Hazard-Mitigating Requirement. *A hazard-mitigating requirement is a functional safety requirement in the sense of [2], which, possibly together with other hazard-mitigating requirements, is part of a mitigation, and thus – when implemented entirely and without error – avoid a hazard or reduce its harmful effects* [2]

The relationship between these terms and concepts is visualized in Fig. 2 by means of the running example from Section 2. As can be seen, the functional requirements of the TCAS from Fig. 1 were subjected to hazard analyses. One hazard that was identified is that the resolution advisory incorporates a descent in low altitude, causing the plane to crash into the ground, potentially resulting in casualties. In this case,

[2] It is to note that hazard-mitigating requirements may themselves cause hazards. Therefore, safety standards (e.g., [6], [7]) demand iterative hazard identification and hazard mitigation.

"Compute Resolution Advisory" from Fig. 1 is a *hazard-inducing requirement* for the *hazard* "RA descend into terrain, causing crash". A possible *mitigation* for this would be to add requirements to achieve the safety goal "Monitor own altitude when computing RA". Following the changes indicated in the mitigation, a *hazard-mitigating requirement* was added which incorporates the barometric altitude (i.e. the altitude measured via air pressure differences): "The TCAS shall not issue DESCEND resolution advisories when barometric altitude is less than 6,000ft".

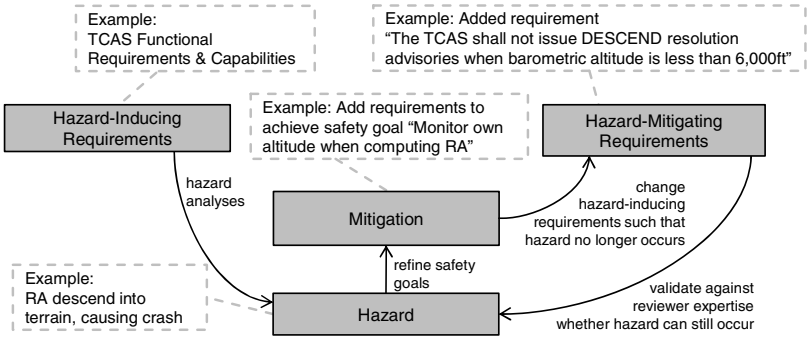

Fig. 2. The Relationship between relevant concepts illustrated through the TCAS Example

4 Supporting Validation through Hazard Relation Diagrams

In Section 4.1, we illustrate necessary development steps before validation of hazard-mitigating requirements can take place. We discuss the concepts needed to conduct validation in Section 4.2 before we introduce the ontology of an integrated representation called Hazard Relation Diagrams in Section 4.3. In Section 4.4 we present a visual notation for Hazard Relation Diagrams by means of the TCAS example.

4.1 Development Steps Prior to Modeling Hazard Relation Diagrams

In order to be able to create Hazard Relation Diagrams which support the validation of hazard-mitigating requirements, development must have progressed sufficiently far. Specifically, in accordance with safety standards (e.g., [6], [7]), the following development steps must have occurred:

Step 1: Functional Requirements Have Been Elicited and Documented by Means of Activity Diagrams. The basis of any development process is eliciting a set of system functions and documenting them by means of functional requirements. Model-based requirements documentation has been seen in the past as a promising avenue to manage complexity not just for safety-critical systems [20]. The advantage of activity diagrams like the one shown in Fig. 1 is that, in embedded systems development, they are particularly suitable to document the functional requirements [24] which are the basis for functional hazard analyses in the next step [5].

Step 2: Hazard Analyses Have Been Conducted. As outlined in Section 1, during early stages of safety assessment, the functional requirements from Step 1 are subjected to hazard analyses [3], [6], [7]. During hazard analyses, a set of hazards will be identified and documented in FHA result tables (see [5]). In particular, for each hazard, its hazard-inducing requirements, trigger conditions, and safety goals are documented. Table 1 shows an excerpt from a FHA of the function *Compute Resolution Advisory (RA)* from Fig. 1. The example hazard from Fig. 2 is shown as hazard H1.

Table 1. FHA of the *Function Compute Resolution Advisory (RA)* from Fig. 1 based on [5]

Req.	Hazard		Effect	Trigger Condition	Safety Goal	
	ID	Description			ID	Description
Compute Resolution Advisory (RA)	H1	Descend into terrain	Impact with terrain, causing crash	Low altitude above ground	SG1	Monitor own current altitude while computing RA
	H2	Climb or descend into traffic trajectory	Fail to avoid intruder, causing collision	Intruder initiates climb or descend	SG2	Monitor intruder's climb or descend rate when computing RA
	H3	No RA issued		Failed to compute CPA	SG3	Announce loss of RA to crew

Step 3: For Each Hazard, Mitigations Are Defined and Documented. In this step, hazard-mitigating requirements are defined and documented which refine the safety goals from Step 2 into concrete mitigations (see, e.g., [6], [7]). This is done by modifying the existing hazard-inducing requirements. For example, the functional requirements from Fig. 1 can be modified by substituting the function *Compute Resolution Advisory (RA)* by a function *Compute Necessary Climb Rate to Achieve Separation Altitude*. Furthermore, a function *Compare with Own Altitude* could be added, which takes into account the current barometric altitude of the own aircraft and compares this information with the computed climb rate. The functional requirements can be modified further such that, if the altitude is less than the 6,000ft, the function *Compute Necessary Climb Rate to Achieve Separation Altitude* is run again with the constraint that the climb rate may not be negative, thereby preventing a descent. If the own current altitude is sufficient to allow descending or if the climb rate is positive, an *RA* can be issued.

4.2 Modeling Concepts for Validation of Hazard-Mitigating Requirements

As outlined above, the mitigations from Step 3 and the hazard-mitigating requirements subsumed therein must be adequate to ensure that the hazards from Step 2 no longer occur during operation. Validation in this sense does not only depend on the knowledge and experience of the reviewers (see, e.g., [16], [17]), but requires specific contextual information about the hazard against which the adequacy of hazard-mitigating requirements must be checked [11]:

- **Hazard.** In order to assess if hazard-mitigating requirements adequately prevent the hazard from occurring during operation (assuming the requirements have been

implemented entirely and without error), it is necessary to have knowledge about what specific hazardous behavior may not occur [10]. Hence, the hazards identified during FHA must be taken into account.

- **Trigger Condition.** A hazard occurs when certain disadvantageous trigger conditions arise during operation [1]. These trigger conditions must be avoided or must be sufficiently unlikely in order for the hazard to be adequately mitigated and must hence also be taken into account.
- **Safety Goal.** Safety goals not only build the basis for safety arguments, they also specify abstract conditions, which must be achieved to mitigate a hazard [8]. Safety goals hence build the basis for mitigation and must be adequately fulfilled.
- **Mitigation.** Mitigations consist of a number of hazard-mitigating requirements, which refine one or more safety goals into concrete, implementable measures to avoid the hazard or reduce its harmful effects (see Definition 2). These must be made explicit in order for reviewers to be able to assess their adequacy.

4.3 Ontology for Hazard Relation Diagrams

In order to foster the validation of hazard-mitigating requirements, we propose integrating the concepts necessary for validation into one diagrammatic representation called Hazard Relation Diagrams. Hazard Relation Diagrams are an extension of UML/SysML activity diagrams, as shown in the ontology in Fig. 3.

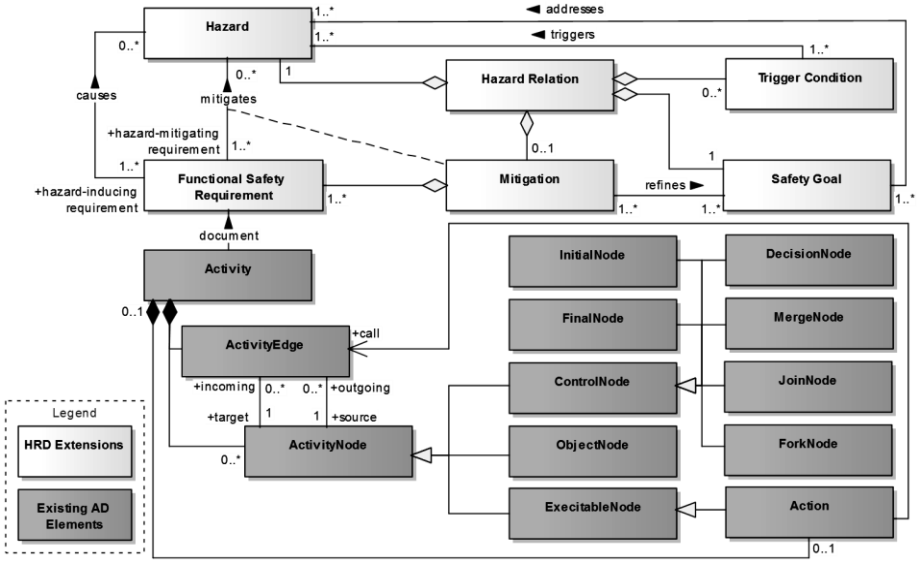

Fig. 3. Ontology for Hazard Relation Diagrams

The modeling constructs from Fig. 3 displayed in dark grey show the excerpt of the UML meta-model for activity diagrams presented in [25]. In order to document the contextual information about hazards, we have extended this excerpt by the modeling

constructs and their relationships introduced in Section 4.2 (displayed in light grey). The resulting ontology shown in Fig. 3 defines the modeling foundation for Hazard Relation Diagrams. In Hazard Relation Diagrams, UML activities are used to document functional safety requirements. These may have two roles: Functional safety requirements can be *hazard-inducing requirements* (see Definition 3) or *hazard-mitigating requirements* (see Definition 4). As can be seen, the core of a Hazard Relation Diagram is a *Hazard Relation* which associates one *hazard* to its set of *trigger conditions*, *safety goals* (which specify what must be achieved to avoid the hazard), and at most one *mitigation*. A mitigation comprises a set of hazard-mitigating requirements (see Definition 2). Section 4.4 proposes a visual notation for this ontology. It is to note that functional requirements for a system are in practice not specified in a single but in multiple activity diagrams, which are furthermore not partitioned with respect to hazards. In consequence, mitigations could comprise hazard-mitigating requirements scattered across multiple diagrams.

4.4 Visual Notation for Hazard Relation Diagrams

We have defined a visual notation for the ontology from Section 4.3, which is shown in Fig. 4. In this case, the hazard mitigation has been defined entirely in a single Hazard Relation Diagram.

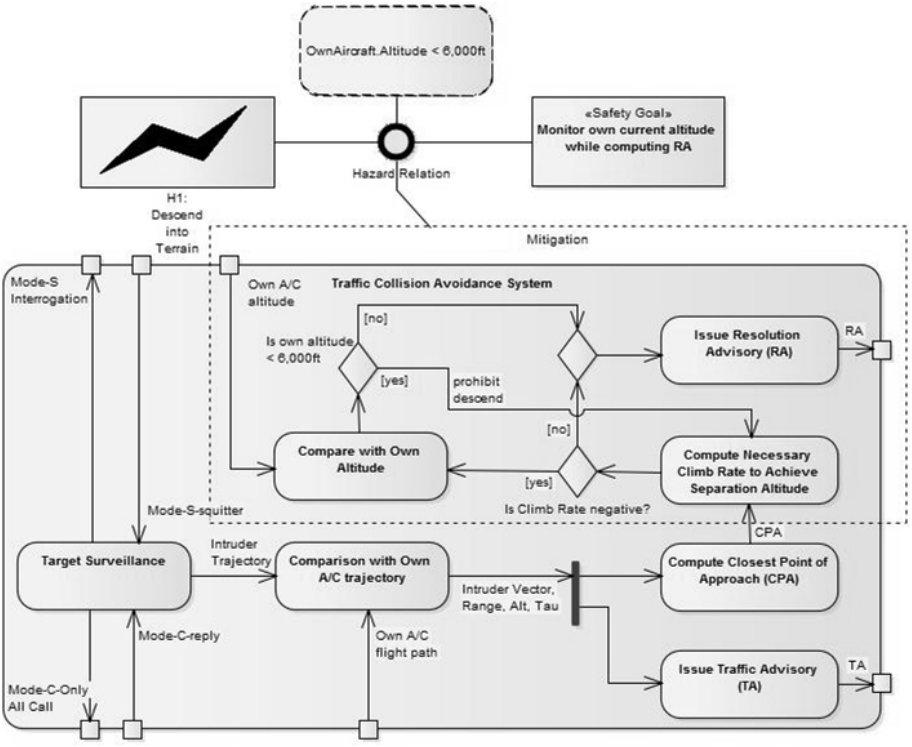

Fig. 4. Hazard Relation Diagram for Hazard H1 from Table 1

As can be seen, *Hazard Relations* are represented by a circle with a thick wall. *Safety goals* are represented as UML classes, stereotyped "<<Safety Goal>>" and containing its description. *Trigger conditions* are represented by dashed rounded rectangle shapes, similar to UML states. The *hazard-mitigating requirements* pertaining to the concrete *mitigation* are surrounded by a dashed partition, which in turn is associated with the Hazard Relation. The key idea of this representation is to focus on the dependencies between the contextual information necessary to validate hazard-mitigating requirements. Specifically, Fig. 4 shows the Hazard Relation Diagram for hazard H1 shown in Table 1. In this Hazard Relation Diagram, the hazard-mitigating requirements from Step 3 in Section 4.1 have been added to the TCAS specification from Fig. 1. As can be seen, instead of simply computing a *RA*, the TCAS will compute (positive or negative) climb rates based on the aircraft's own barometric altitude, thereby fulfilling the safety goal. During review, however, it may turn out, that the hazard-mitigating requirements contained in the mitigation are inadequate: A reviewer may notice that especially above mountains, the barometric altitude, i.e. the altitude measured through air pressure, can differ dramatically from the aircraft's altitude above ground. This means the trigger condition of the hazard is not avoided and may still cause the hazard during operation.

It is to note that in this simple example, only one mitigation for one hazard has been considered. In practice, it could be the case that there are several hazards which are addressed by the same mitigation. Similarly, multiple candidate mitigations could exist that present alternatives to address the same hazard (possibly with varying degrees of adequacy). The purpose of Hazard Relation Diagrams is to direct the attention of a reviewer on the adequacy of one mitigation with respect to one hazard. Therefore, it is necessary to review each candidate mitigation individually with respect to the corresponding hazard. In addition, it may often be the case that there are a number of hazard-mitigating requirements scattered across different activity diagrams (or, in case of large activity diagrams, in different positions within the same diagram). In this case, a Hazard Relation Diagram includes multiple mitigation partitions which surround any and all hazard-mitigating requirements that collectively make up the mitigation, thereby possibly aggregating several activity diagrams.

5 Impact of Hazard Relation Diagrams on Reviews

To investigate our hypothesis that by using Hazard Relation Diagrams, reviewers are more likely to base their adequacy judgment on contextual information about the hazard, we designed an empirical study, which is explained in Section 5.1. Section 5.2 reports on preliminary findings from a pilot test aimed to validate the study design. Section 5.3 reports on threats to validity.

5.1 Experiment Design

We designed a one-way between-subjects experiment [26]. We specifically opted for a between-subjects design because a repeated measures design would have significantly increased training overhead as participants needed instruction on a number of topics, i.e. safety engineering fundamentals, Hazard Relation Diagrams, and Functional Hazard Analysis. We therefore divided participants into treatment and control

groups, where the treatment group was asked to perform a review of hazard-mitigating requirements by means of Hazard Relation Diagrams. The control group was asked to perform the review based on activity diagrams and FHA result tables.

Experimental Material. The experimental material consisted of a model-based requirements specification like the one in Fig. 1. The example system was that of an automotive Adaptive Cruise Control (ACC). The specification consisted of one activity diagram comprising five hazard-inducing requirements, for which a FHA was conducted (see Section 4.1). We specifically opted for a system from the automotive domain as in contrast to the TCAS example from Section 2, as participants are assumed to be familiar with automotive systems and hence required less instruction. The FHA yielded a total of ten hazards. Five of these hazards were randomly selected and *adequately* mitigated. To do so, for each hazard, a variation of the activity diagram was derived in which hazard-mitigating requirements have been documented that will avoid the hazard during operation. The remaining five hazards were *inadequately* mitigated. To do so, for each hazard, a variation of the activity diagram was derived in which hazard-mitigating requirements have been documented which contain semantic mistakes allowing the hazard to still occur during operation. For the treatment group, each adequate and inadequate activity diagram was extended into a corresponding Hazard Relation Diagram similar to Fig. 4.

Measurements. We measured a number of variables, including time needed to complete each review as well as the number of correctly and incorrectly assessed adequate and inadequate mitigations. In addition, we measured the self-reported confidence of a participant in the assessment for each review as well as several items from the Technology Acceptance Model 3 (TAM3, [27]) and Task Technology Fit (TTF, [28]) questionnaire. These self-report items were measured on a 5-point-Likert scale. Furthermore, participants were asked to issue a brief written rationale stating why participants assessed some mitigation as either adequate or inadequate.

Procedure. We conducted a pilot test in order to validate the experimental material as well as the experimental design. Before the pilot test, a short briefing was administered which instructed participants on how to perform the reviews. The order in which the ten hazard mitigations were presented was randomized for each participant to reduce primacy, recency, and carry-over effects [26]. In order to ensure that both participant groups reviewed approximately equally many information items, an effort was made to present only the one row from the FHA result relevant to one hazard to the control group participants at the time. The experimental procedure consisted of the following steps, as shown in Fig. 5:

- **Step 1: Introduction, Informed Consent, & Demographics.** The pilot test began with a brief introduction, where informed consent as well as demographic data was collected and participants were informed of their option to discontinue at any point.
- **Step 2: Separation into Groups.** Based on the demographic information, participants were then randomly assigned into treatment group and control group. An effort was made to distribute participants such that an equal number of participants with corresponding experience levels were assigned to each group.

- **Step 3: Instructions & "Dry Run".** Both groups were presented with instructions on how to review the experimental material once again. Furthermore, two example runs were performed in which the participants could rehearse the review task.
- **Step 4: Review of Hazard Mitigation & Self-Report Confidence.** Participants were asked to review hazard mitigations pertaining to one randomly selected hazard. Together with the subsequent step, this step was hence repeated ten times. For each randomly selected hazard, participants could review the experimental material for an indeterminate amount of time and indicate "yes" if they are of the opinion that the hazard may still occur during operation and "no" otherwise. In addition, participants were asked to rate their own confidence. Participants were able to change their assessment and self-reported confidence as often as they wished.
- **Step 5: Self-Report Rationale.** Participants were asked to state a brief reason why they chose "yes" or "no" in the previous step. This rationale was used by the experimenters to draw conclusions about the decision making process and assess what information was used to make the adequacy judgment. Participants were given the opportunity to return to the previous step and change their answer if thinking about the rationale made them change their mind. Furthermore, the experimental material along with their decision from the previous step was shown for reference.
- **Step 6: Post-Hoc Questionnaire.** Both groups were presented with several items from the TAM3 [27] and TTF [28] in order to gain data on the participants' experience during review, perceived usefulness of the respective notation, and the respective notation's ability to assist in the review process.

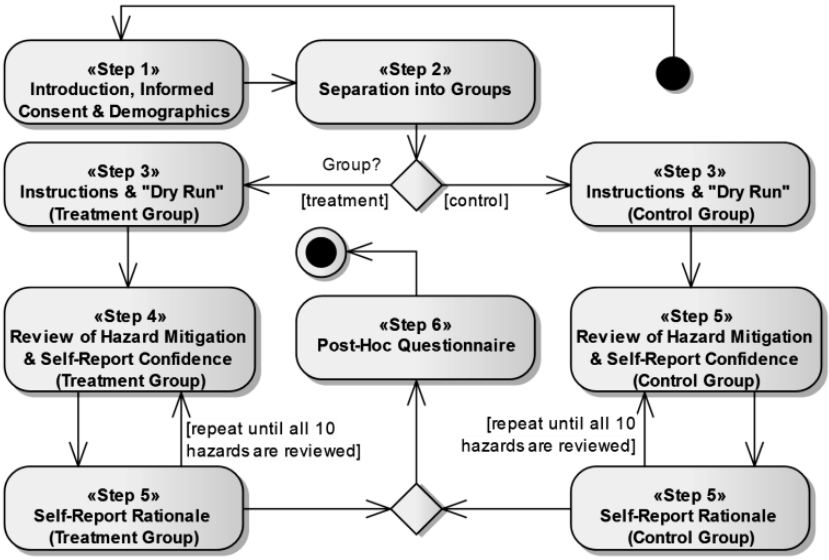

Fig. 5. Overview over the Experimental Procedure

Participants. Participants in the pilot test were researchers from the authors' research group. Most participants possessed or were in the process of obtaining a Master's or PhD degree in software engineering; only one participant already possessed a doctoral

degree. All participants self-reported experience levels in requirements engineering and conceptual modeling from academic and industrial research projects and experience in static quality assurance from their daily academic work. A total of ten participants ranging from 25 to 36 years of age completed the study, yielding n = 5 per group. Albeit gender is not assumed to influence the experiment results, it is to note that no female participant participated in the study.

5.2 Preliminary Results from the Pilot Test

At the time of writing this manuscript, a detailed analysis of some of the measurements from Section 5.1 (e.g., analysis of correctness of participants' responses as well as an evaluation of TAM3 and TTF results) was still underway. Furthermore, the purpose of the pilot study was to validate the experimental design as well as the experimental material. In the following, we hence present findings regarding the suitability of design and material as well as some preliminary results on the impact of Hazard Relation Diagrams on reviews.

Insights regarding the Experimental Design. Experience from the pilot test indicates that training was a relevant concern, as participants were unfamiliar with the specific input/output-pin oriented notation of activity diagrams (cf. Fig. 1). During informal post-experimental discussions, one participant indicated that albeit he was familiar with this notation, he felt that more rigorous instruction about the specific semantics would have increased his understanding. In addition, participants indicated that the concept of *hazards* and *mitigations* were hard to understand in the pre-experimental briefing and introduction step. In consequence, more emphasis must be placed on explaining the used notations in detail. However, these findings also validate our choice of using a between-subjects design as it is likely that additional instruction in multiple notations will result in a much steeper learning curve.

Insights regarding the Experimental Material. Results show that the example system was generally well understood by participants. This confirmed our assumption that an example of the automotive industry is suitable for the purpose of the experiment. However, results also show that some example diagrams were somewhat ambiguous. This can be seen from the fact that rationales regarding the adequacy judgment differed significantly between participants. For example, one hazard mitigation was designated inadequate by the experimenters because it specified that a signal shall be considered during the operation of the ACC which is not available on any input pin. The reasons given by participants for their judgment varied considerably: Some participants stated the reason designated by the experimenters, yet others argued based on ambiguity in the execution order or based on syntactic flaws in the diagram. These findings show that albeit the example is suitable, some hazard mitigations must be revised in order to reduce variation in diagram comprehension.

Insights regarding Review Objectivity. We conducted a qualitative analysis of the rationales reported by the participants. This was done by reading the rationales given for each hazard mitigation (both adequate and inadequate) and classifying them with regard to the referenced information in the rationale. Specifically, we differentiated

rationales being based on semantic properties (e.g., if a requirement is factually wrong), syntactic properties (e.g. if there was a syntactic mistake in the diagram), trigger conditions (e.g., if all trigger conditions were successfully avoided), and safety goals (e.g., if the safety goal was properly fulfilled). Results indicate that in the control group, out of a possible 50 rationales (five participants times ten hazards), 41 rationales were given. Of these 41 rationales, only six referenced trigger conditions. More strikingly, four out of these six rationales were given by the same participant. No rationale given by the control group referenced safety goals and one rationale stated syntactic mistakes. The majority of 34 stated semantic reasons. In contrast, in the treatment group, a total of 45 rationales were given. Of these 45 rationales, eleven referenced trigger conditions and six argued on the basis of safety goal fulfillment. Interestingly, all except one participant referenced trigger conditions at least once. Three rationales referenced syntactic mistakes and 25 rationales were based on semantic reasons. These results show that while the treatment group based their rationale more often on contextual information, the control group based their judgments almost entirely on activity diagram semantics, which supports our hypothesis.

5.3 Threats to Result Validity

As with any study, some threats to validity which impair the ability to draw conclusions from the experimental results remain. These are discussed in the following.

Internal and Construct Validity. One critical issue for the study at hand is the suitability of the experimental procedure and materials. To increase internal and construct validity, we conducted a pilot test. The pilot test yielded a few issues that warrant revisions to the experimental procedure and material, as outlined in Section 5.2.

Conclusion Validity. Only ten participants participated in the pilot test. In this case, such a small number of participants yield insufficient statistical power. Therefore, we emphasize that the results reported in Section 5.2 are preliminary and give mere indications in support of our hypothesis. Further testing with larger numbers of participants is expected to produce additional results and is subject of ongoing work.

External Validity. Neither participant population of the pilot nor experimental material is representative. Therefore, repetition studies with different populations and case studies with industry representatives ought to be carried out to ensure generalizability of results.

6 Related Work

A comprehensive overview over validation techniques that can be used to validate requirements is given in [29]. Albeit a number of techniques have been proposed to foster validation (e.g., inspection-like techniques [30], reading techniques [31], or prioritization techniques [32]), in practice, unstructured reviews remain the main vehicle to assess the adequacy of any type of requirement [24]. Furthermore, review techniques are typically generic in nature and support the developers in assessing

requirements without particular focus on hazards. The methodology proposed in [29] improves upon this issue by combining in a joint process the identification of hazards and necessary changes to the requirements in order to mitigate the hazards. Formal quality assurance approaches such as [33], [34] place particular emphasis on ensuring that mitigations are formally correct. Such approaches focus on verifying system behavior and system design against behavioral constraints (e.g., real-time requirements).

A number of approaches propose joint safety/security processes, allowing for combined threat and hazard identification and resolution (e.g., [35], [36]). Most notably, misuse cases (e.g., [36], [37]) have been applied to safety in order to identify unsafe interaction between the system and its context and find candidate interactions to resolve them. Moreover, security threat analysis, safety hazard identification and subsequent requirements derivation have often been proposed to occur in a mutually beneficial manner (see [38] for a comprehensive comparison of techniques).

Goal-oriented (e.g., [39]) and scenario-based (e.g. [4]) approaches allow eliciting a set of safety requirements with regard to system hazards. For example, the KAOS approach [39] provides provably correct refinement patterns that can be used to refine hazard obstructions, i.e. obstructions that may arise during operation which impair safety-critical goal satisfaction, into operationalization resolving them. However, while this leads to a formally correct specification, whether or not the specification is adequate with regard to the semantic domain requires additional validation [40].

A wide range of non-goal-based formal approaches such as [35], [41], [42], [43], or [44] have been proposed. These require that at least some portion of the system has already been designed or even implemented and focus on analyzing timing constraints [44], behavioral constraints [35], design invariants [43], or event-based failure propagation [41] and can be used to deduce requirements which circumvent certain hazardous conditions. Hence, safety requirements that can be elicited using these approaches are more akin to technical constraints that become apparent during late development states rather than early phase requirements.

7 Conclusion and Outlook

In this paper, we have proposed a diagrammatic representation called Hazard Relation Diagrams which can be used during validation of hazard-mitigating functional safety requirements. Hazard Relation Diagrams combine functional requirements intended to mitigate a specific hazard with contextual information about the hazard, i.e. safety goals to be achieved and triggering conditions during operations under which the hazard occurs. We have shown how Hazard Relation Diagrams can be used during development and we have argued that using Hazard Relation Diagrams leads to increased consideration of contextual information when validating safety requirement adequacy by means of reviews. We have outlined an empirical study designed to investigate this claim and shown results from a pilot test. Findings indicate that participants using Hazard Relation Diagrams based their judgments more often on contextual information than the control group. Future work will entail revising instructional and experimental material and continue with the empirical investigation. Specifically, it is planned to carry out the experiment with students from an undergraduate requirements engineering course and students from a graduate quality assurance course.

Acknowledgments. This research was partly funded by the German Federal Ministry of Education and Research under grant number 01IS12005C. We thank Arnaud Boyer of Airbus Defence and Space for his consultation regarding TCAS and FHA and Dr. Kai Petersen, Marian Daun, and André Heuer for their feedback on the study design.

References

1. Leveson, L.: Safeware: System Safety and Computers. Addison-Wesley, Boston (1995)
2. Firesmith, D.: Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements. J. Obj. Tech. **3**, 27–42 (2004)
3. Leveson, L.: Engineering a Safer World. MIT Press, Boston (2011)
4. Allenby, K., Kelly, T.: Deriving safety requirements using scenarios. In: Proc. 5th Int. Symp. Requirements Eng., pp. 228–235 (2001)
5. Ericson II, C.: Hazard Analysis Techniques for System Safety. Wiley, New York (2005)
6. ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. SAE International (1996)
7. ISO26262: Road Vehicles - Functional Safety. International Organization for Standardization (2011)
8. Bishop, P., Bloomfield, R., Guerra, S: The future of goal-based assurance cases. In: Proc. Workshop on Assurance Cases, pp. 390–395 (2004)
9. Whyte,D.: Moving the goalposts: the deregulation of safety in the post-piper alpha offshore oil industry. In: Proc. 47th Ann. Conf. Political Studies Assoc. of the UK (1997)
10. Hatcliff, J., Wayssyng, A., Kelly, T., Comar, C., Jones, P.: Certifiably safe software-dependent systems: challenges and directions. In: Proc. Future Softw. Eng., pp. 182–200 (2014)
11. Boehm, B.: Verifying and Validating Software Requirements and Design Specifications. IEEE Softw. **1**, 75–88 (1984)
12. Glinz, M.: Improving the quality of requirements with scenarios. In: Proc 2nd World Cong. Softw. Qual., pp. 55–60 (2000)
13. ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International Organization for Standardization (2011)
14. Boehm, B.: Software Engineering Economics. Prentice Hall, Englewood Cliffs (1981)
15. Glinz, M., Fricker, S.: On Shared Understanding in Software Enigneering. Computer Sci. Res. Dev. doi:10.1007/s00450-014-0256-x
16. Gacitua, R., Ma, L., Nuseibeh, B., Piwek, P., de Roeck, A., Rouncefield, M., Sawyer, P., Willis, A., Yang, H.: Making tacit requirements explicit. In: Proc. 2nd Int. Workshop on Manag. Req. Knowl., pp. 85–88 (2009)
17. Lisagor, I., Sun, L., Kelly, T.: The illusion of method: challenges of model-based safety assessment. In: Proc. 28th Int. Syst. Safety Conf. (2010)
18. Sun, K.: Establishing Confidence in Safety Assessment Evidence. Dissertation, Univ. of York (2012)
19. Flynn, D., Warhurst, R.: An Empirical Study of the Validation Process within Requirements Determination. Inf. Syst. J. **4**, 185–212 (1994)
20. Davies, I., Green, P., Rosemann, M., Idulska, M., Gallo, S.: How do Practitioners use Conceptual Modeling in Practice? Data & Knowl. Eng. **58**, 358–380 (2006)
21. US FAA: Introduction to TCAS II, Version 7.1. (2011). http://goo.gl/EPCYzI
22. Glinz, M.: On non-functional requirements. In: Proc. IEEE Int. Requirements Eng. Conf., pp. 21–26 (2007)
23. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional Requirements in Software Engineering. Kluwer, Boston (2000)

24. Sikora, E., Tenbergen, B., Pohl, K.: Industry Needs and Research Directions in Requirements Engineering for Embedded Systems. Requirements Eng. **17**, 57–78 (2012)
25. Störrle, H.: Semantics of UML 2.0 activities. In: Proc. IEEE Symp. Visual Lang. and Human-Centric Computing, pp.235–242 (2004)
26. Wohlin, C., Runeson, P., Höchst, M., Ohlson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering – An Introduction. Springer, Heidelberg (2012)
27. Venkatesh, V., Bala, H.: Technology Acceptance Model 3 and a Research Agenda on Interventions. Decision Sci. **39**, 273–315 (2008)
28. Goodhue, D.: Development and Measurement Validity of a Task-Technology Fit Instrument for User Evaluations or Information Systems. Decision Sci. **29**, 105–138 (1998)
29. Denger, C.: SafeSpection — A Framework for Systematization and Customization of Software Hazard Identification by Applying Inspection Concepts. Dissertation, TU Kaiserslautern (2009)
30. Fagan, M.: Design and Code Inspections to Reduce Errors in Program Development. IBM Syst. J. **3**, 182–211 (1976)
31. Shull, F., Rus, I., Basili, V.: How Perspective-Based Reading Can Improve Requirements Inspections. IEEE Computer **33**, 73–79 (2000)
32. Li Q, Boehm B, Yang Y, Wang Q: A value-based review process for prioritizing artifacts. In: Proc. Int. Conf. Softw. Syst. Process, pp. 13–23 (2011)
33. Bitsch, F.: Safety patterns - the key to formal specification of safety requirements. In: Voges, U. (ed.) SAFECOMP 2001. LNCS, vol. 2187, pp. 176–189. Springer, Heidelberg (2001)
34. Heitmeyer, C., Kirby Jr., J., Labaw, B., Archer, M., Bharadwaj, R.: Using Abstraction and Model Checking to Detect Safety Violations in Requirements Specifications. IEEE TSE **24**, 927–948 (1998)
35. Zafar, S., Dromey, R.: Integrating safety and security requirements into design of an embedded system. In: Proc. 12th Asia-Pacific Soft. Eng. Conf., pp. 1530–1362 (2005)
36. Sindre, G.: A look at misuse cases for safety concerns. In: Ralyté, J., Brinkkemper, J., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences. IFIP, vol. 244, pp. 252–266. Springer, Heidelberg (2007)
37. Katta, V., Raspotnig, C., Karpati, P., Stålhane, T.: Requirements management in a combined process for safety and security assessments. In: Proc. Intl. Conf. on Availability, Rel., and Security, pp. 780–786 (2013)
38. Raspotnig, C., Opdahl, A.: Comparing Risk Identification Techniques for Safety and Security Requirements. J. Sys. Softw. **86**, 1124–1151 (2013)
39. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, New York (2009)
40. Letier, E., van Lamsweerde,A.: Deriving operational software specifications from system goals. In: Proc. Future of Softw. Eng., pp. 119–128 (2002)
41. Belli, F., Hollmann, A., Nissanke, N.: Modeling, analysis and testing of safety issues - an event-based approach and case study. In: Saglietti, F., Oster, N. (eds.) SAFECOMP 2007. LNCS, vol. 4680, pp. 276–282. Springer, Heidelberg (2007)
42. Chen, Z., Motet, G: Formalizing safety requirements using controlling automata. In: Proc. 2nd Int. Conf. Depend., pp. 81–86 (2009)
43. Guillerm, R., Sadou, N., Demmou,H.: Combining FMECA and fault trees for declining safety requirements of complex systems. In: Proc. Ann. Europ. Safety and Rel. Conf., pp. 1287–1293 (2011)
44. Hansen, K., Ravn, A., Stavridou, V.: From Safety Analysis to Software Requirements. IEEE TSE **24**, 573–584 (1998)

http://www.springer.com/978-3-319-16100-6

Requirements Engineering: Foundation for Software Quality
21st International Working Conference, REFSQ 2015, Essen, Germany, March 23-26, 2015. Proceedings
Fricker, S.A.; Schneider, K. (Eds.)
2015, XIV, 333 p. 63 illus., Softcover
ISBN: 978-3-319-16100-6