A Study of Safety Patterns: First Results

Mario Gleirscher, Stefan Kugele

TUM-I1640

A Study of Safety Patterns: First Results

Mario Gleirscher and Stefan Kugele

Institut für Informatik, Technische Universität München, Germany
{gleirsch,kugele}@in.tum.de

Abstract. We consider the reuse of knowledge in the analysis, design, and assurance of safety-critical appliances. We refer to this knowledge as *safety patterns* for whom a considerable amount of literature has been published. However, relationships across this literature seem relatively unexplored. Our first analysis indicates, e.g. scattered information about properties and combinations of such patterns. In this paper, we present results of the first stage of a three-staged literature study. With these results we aim to stimulate discussion and request for comments for the subsequent stages we are going to conduct—a systematic mapping study and a systematic literature review.

Keywords: System safety, engineering, patterns, reuse, literature study.

1 Introduction

Safety is an indisputably critical property of engineered systems and their control software. The analysis and assurance of this property and related properties such as reliability and availability remains a major activity throughout the whole life cycle of these systems. The general procedure and many of the key elements of a safety engineering effort are largely reflected in canonical methods, techniques, and standards. Moreover, in the last four decades, a lot of literature has been published on *reusable knowledge in safety engineering* and related fields such as reliability engineering. We refer to this knowledge by the term *safety patterns*, i.e., analysis, design, and assurance patterns for safety concepts or measures to be taken in the construction and operation of engineered systems.

Safety patterns are a concept known in many of the collaborating branches of safety engineering. However, our first analysis indicates that an overview of (i) properties and combinations of safety patterns and (ii) the variety of reuse in this area is not sufficiently reflected in the scientific literature. We consider a *cross-disciplinary literature study* as important – to improve the understanding of (i) and (ii), – to foster the relationships between recent safety analysis and design approaches, and – to identify potentials for enhancement and research gaps. Hence, we rephrase this challenge into the *research question*:

> What can we say about the (i) key elements and (ii) the state-of-the-art of reusable knowledge in system safety engineering?

With this study, we aim at a contribution to the consolidation of findings about the reusable fragment of safety engineering knowledge as well as to the practical safety engineering body of knowledge.

Related Work. During our literature search we only found one relevant publication, the one of Preschern et al. [17]. They provide a survey of 12 safety-related methods using patterns and point out that they could not find a comparable study on safety patterns and their application. The authors discuss these methods regarding their target domain, involved types of patterns (i.e., process, safety tactics, architecture, timing), and degree of detail. They put a strong focus on pattern application and use in a safety process. However, as their study is not systematic in the sense of, e.g. Kitchenham [8], it is difficult to estimate literature coverage and to conduct a replication.

Outline. In the following, we give an overview of the design of our whole study (Sect. 2) and show how we approached its first stage (Sect. 3). In Sect. 4, we discuss works selected in the first stage by giving an annotation. Section 5 reflects on our results and provides insight into our next steps.

2 Three-staged Study Design

We organize our study into three consecutive and coordinated stages:

In the **first stage**, we aim at an *overview of adequate search terms, structuring criteria, and works of the field.* We address this goal by an *annotated bibliography* [9]. Our discussion is based on annotations of selected publications.

In the **second stage**, we focus on *gaining an exhaustive overview of the field, and determine research directions, trends and potential gaps.* We want to achieve this goal by a *systematic mapping study* [13].

In the **third stage**, our goal is to *go in-depth into the most relevant publications and answer specific research questions.* For this, we want to carry through a *systematic literature review* [8]. We aim at understanding the relationships between the various safety pattern approaches, properties of safety patterns and combinations thereof, and relationships with reliability and dependability patterns. For each pattern, we want to answer questions motivated by the main question described in Sect. 1: – Where did the pattern originate from? – Which role does the pattern have? In which domains is it applied? – Which technologies does the pattern abstract from? – Which notion of safety underlies the pattern? – Which properties does the pattern have? – Which (formal) model is used to explain the pattern?

We present our approach and results of the first stage in Sects. 3 and 4.

3 Approach of the First Stage

Search and Selection Results. We briefly describe how we selected the works annotated in Sect. 4. Based on a discussion of what we understand by the term *safety pattern* and based on our knowledge of the field, we crafted *search strings* for two databases, see Table 1.

Google Scholar: To reduce the number of results to 50-200, the string looks for `"safety pattern"` and at least one out of 11 terms typical for the field. This search interface constrains the input length; we left out `"hardware pattern"`.

Scopus: Because of fewer results, we could relax the string to look for a combination of one out of five relevant terms containing `"safety"` and one out of seven relevant terms containing `"pattern"`.

Table 1. Search settings and results by Dec 30 2015. Legend: p... publications, d... duplicates.

Google Scholar		Elsevier Scopus

`"safety pattern"`
`("safety concept" OR`
`"safety measure" OR`
`"safety mechanism" OR`
`"safety system" OR`
`"safety function"`
`OR`
`"analysis pattern" OR`
`"requirements pattern" OR`
`"modeling pattern" OR`
`"design pattern" OR`
`"architecture pattern" OR`
`"code pattern")`
		`ALL(`
		`("safety concept" OR`
		`"safety measure" OR`
		`"safety mechanism" OR`
		`"safety system" OR`
		`"safety function")`
		`AND`
		`("analysis pattern" OR`
		`"requirements pattern" OR`
		`"modeling pattern" OR`
		`"design pattern" OR`
		`"architecture pattern" OR`
		`"code pattern" OR`
		`"hardware pattern"))`

Anywhere in the article, without patents.		Only in title, abstract, and meta-data, without patents.
Procedure:		
Search:	$91p - 4d$ +	$78p - 1d = 164p$
1st rating:	$27p - 2d$ +	$6p = 31p$
1st rating & disc.:	$30p - 2d$ +	$8p = 36p$
2nd rating & disc.:	$9p - 1d$ +	$1p = 9p$
Final selection:		**9 publications**

After *duplicate removal*, we filtered our results (164 publications) in two rating steps:

First, based on the aforementioned discussion, each researcher individually (i) considered the title, abstract, and conclusion of each publication, and (ii) rated it by 1 (for inclusion) and < 1 (for exclusion). We gained an agreement level of 91%. To qualify this, we applied Cohen's κ which is defined as $\kappa = (p_0 - p_e)/(1 - p_e)$, with the determined percentage agreement p_0 and the hypothetical probability of chance agreement p_e, using the observed data to calculate the probabilities of each rater randomly for each category, i.e., inclusion and exclusion.

We gained $p_0 = 0.91$, $p_e = 0.64$, and $\kappa = \mathbf{0.75}$, see Table 2. According to [6], $0.61 \leq \kappa \leq 0.8$ corresponds to a *substantial* agreement.

Table 2. Cohen's κ

		Rater 1	
		Inc	Exc
Rater 2	Inc	31	6
	Exc	9	118

Discrepancies between the raters were discussed and only publications whose added ratings equaled 2 were included. We *excluded* a publication if it complied with at least one of the following criteria: It is

– out of context, too general or lacks detail, methodology or technical aspects of safety engineering,
– focused on an application example or a case study,
– not the best fit out of several works published by the same author(s),
– a master or PhD thesis, a collection or book, or a technical report (as we aim to select a low number of publications for this first stage),
– not written in English or German (as we do not speak other languages),
– unavailable via our library services or electronically.

Second, we individually rated the residual 36 publications by 0, 1, 2, and 3 (for strong and weak exclusion, weak and strong inclusion) and included only publications whose added ratings were ≥ 3 after a discussion of discrepancies. 10 publications were left to review in-depth by at least one researcher. We identified one false positive[1] such that 9 remained for our annotated bibliography and further discussion. This procedure is summarized in Table 1.

Analysis and Classification. We identified and applied the following criteria to classify the publications and patterns we found:

– *supported engineering step*, e.g. requirements management (R), design (D), and implementation (I),
– *category*, e.g. procedural pattern (PP, process-based, e.g. for ensuring criticality assignment), transformation pattern (TP), property specification pattern (SP, e.g. for specifying contracts), (architectural) design pattern (DP, e.g. for fault tolerance or containment, criticality, and separation; building blocks in computer-aided engineering), argumentation pattern (AP),
– *abstraction*, i.e., one or more out of, e.g. software (SW), electronic hardware (HW), mechanics (M),
– applied *(formal) method or language* if any, e.g. structured textual requirements (STR), state machine modeling (SM), goal structuring notation (GSN), unified/systems modeling language (UML/SysML).

In Tables 3 and 4, we show the result of our characterization of the 9 reviewed works on safety patterns.

4 Annotation of Selected Works

According to [9], we discussed our condensed search results by (i) a summary and a note on why these works are relevant for our study, as well as (ii) a note on how well these works addressed our expectations on safety patterns.

4.1 Patterns in Argumentation

Argumentation patterns are used to *justify* an acceptable level of safety and to establish *confidence* for safety arguments.

[5] R.D. Hawkins and T.P. Kelly. "Software safety assurance – what is sufficient?" In: 4th IET Int. Conf. Systems Safety, incorporating the SaRS Annual Conference. *Oct. 2009, pp. 1–6*

This paper proposes a way to establish *confidence* for software safety arguments. Certainty about all claims made by a software safety argument cannot be reached due to incomplete information. However, sufficient confidence about the claims has to be achieved. Assurance deficits arising from residual uncertainties have to be made explicit to reason about them. The six steps mentioned in [7] are extended by identifying assurance deficits using a deviation-style analysis (i.e., HAZOP) which examines how uncertainties can be introduced in these steps.

[1] We excluded the work of Gawand, Mundada, and Swaminathan [3], because lack of clarity in both, their notions and pattern discussion, made it difficult for us to establish a well-grounded relationship to our study.

For each assurance deficit, the importance and impact has to be determined (e.g. "intolerable", "broadly acceptable", or "tolerable"). After commenting on the safety argument patterns of Kelly, Weaver, and Ye, the authors present a safety argument pattern catalogue consisting of 5 patterns (see Table 4) tailored to software. GSN with an extension for using parameters is used to describe these patterns.

The authors do not provide a detailed example which made it difficult for us to follow their approach and to evaluate its feasibility. Moreover, their website containing pattern information was not reachable during our investigation.

[12] Robert Palin and Ibrahim Habli. "Assurance of Automotive Safety – A Safety Case Approach". English. In: Computer Safety, Reliability, and Security. *Ed. by Erwin Schoitsch. Vol. 6351. LNCS. Berlin Heidelberg: Springer, 2010, pp. 82–96*

This article aims at demonstrating how automotive safety cases can be produced to justify acceptable system safety. A hierarchical catalogue consisting of a high-level vehicle safety argument and 12 low-level patterns (see Table 4) is presented in detail. The discussion takes place without using a standard set of attributes as opposed to [4]. However, the mentioned patterns are related to each other using the relationships "supported by" and "in context of". GSN extended by *patterns* and *modules* is used to describe the patterns. Their approach is illustrated using an automotive start/stop system for whom the *risk management argument pattern* and the *risk mitigation argument pattern* are instantiated.

The authors claim that their pattern catalogue, based on industry-driven research, facilitates reuse of arguments and supports integration of design and safety activities during development. It would be interesting to see more evidence for this valuable claim.

4.2 Patterns in Requirements Management

Requirements patterns encompass procedures to *decompose* requirements as well as *textual templates* to write structured requirements.

[1] Pablo Oliveira Antonino and Mario Trapp. "Improving consistency checks between safety concepts and view based architecture design". In: 12th Probabilistic Safety Assessment & Management Conference (PSAM), Honolulu, Hawaii, USA. *Techno-Info Comprehensive Solutions, 2014*

This article tackles the problem of inconsistencies between safety concepts and automotive architecture designs. These inconsistencies can occur when safety requirements are changed during development but the related safety concepts are not. The authors argue that it is hardly possible to keep safety assurance artefacts (e.g. safety concepts as defined in ISO 26262) up to date and safety concepts consistent with the architecture. Antonino and Trapp propose a technique for specifying safety concepts. It consists of (i) a *safety concept decomposition pattern* and (ii) a *parametrized safety concept specification template*. For the former, a meta-model is proposed and all concepts are explained. This meta-model helps structure a safety concept specification (according to ISO 26262). The latter is a catalogue of parametrized safety concept specifications which helps write down requirements following a defined syntax (structure). To explain the concepts of their meta-model, a UML class diagram is used. A Backus-Naur-like notation is used to describe the templates. Both concepts are linked for traceability. An example of a power sliding door module shows their approach.

We consider it important to add further arguments as to how the described set of templates is complete.

[2] Pablo Oliveira Antonino et al. "The Safety Requirements Decomposition Pattern". In: Computer Safety, Reliability, and Security*. Ed. by Floor Koornneef and Coen van Gulijk. Vol. 9337. LNCS. Berlin Heidelberg: Springer, 2015, pp. 269–82.* ISBN*: 978-3-319-24254-5*

The authors propose a *procedural pattern* for decomposing safety requirements such that *traceability* to the architectural design and to a fault propagation model (i.e., *fault trees*) is established to perform complete, consistent and early hazard mitigation. The discussion takes place without using a standard set of attributes as opposed to [4].

Antonino et al. use *conceptual modeling* based on UML class diagrams to characterize safety requirements, functional and technical architecture, faults and traceability. Both architectural levels are also described by UML class diagrams. *Fault trees* are employed for analyzing fault propagation across the architecture, though, without reference to a behavioral model. Restriction to a structural model can lower the precision of a traceability analysis. It would be interesting to delve into the relationship of their approach to GSN and safety cases (e.g. as shown in [12]).

[11] Markus Oertel, Omar Kacimi, and Eckard Böde. "Proving Compliance of Implementation Models to Safety Specifications". English. In: Computer Safety, Reliability, and Security*. Ed. by Andrea Bondavalli, Andrea Ceccarelli, and Frank Ortmeier. Vol. 8696. LNCS. Berlin Heidelberg: Springer, 2014, pp. 97–107.* ISBN*: 978-3-319-10556-7*

This article discusses an automated approach to show the compliance of an implementation to safety requirements. The authors argue that the current manual process is error prone and time consuming. To automate this process, safety requirements need to be formalized. For this, they use *safety contracts*. These contracts consist of *assumptions and promises*, both expressed as safety property specification patterns. According to Sect. 3, we classify them as structured textual requirements. These textual building blocks are translated into the linear-time temporal logic LTL to be checked with the VIS model checker. The implementation is done using MATLAB/Stateflow.

Aside from the requirements specification, the authors systematically inject faults into their models. This way, they can analyze which combination of faults results in a violation of a requirement, thereby resembling fault tree analysis. Oertel, Kacimi, and Böde demonstrate their approach using an automotive light manager system.

The safety contracts are discussed in sufficient detail, but the process of obtaining fault trees from computed cut-sets was hard to follow for us due to a lack of technical information.

4.3 Patterns in Architectural Design

Architectural design patterns aim at providing reusable engineering knowledge to increase system safety.

[4] Lars Grunske. "Transformational patterns for the improvement of safety properties in architectural specification". In: Proc. 2nd, 3rd and 4th Nordic Conf. On Pattern Languages Of Programs (VikingPLoP)*. 2003, pp. 135–50*

The author discusses patterns for transforming a system architecture design into a behaviorally equivalent design which is said to better address associated safety requirements, i.e., by reducing probabilities of hazards. The patterns aim at handling three types of component failures: unavailability, incorrect reactions, and timing deviations. The article describes 3 patterns for *design-time fault prevention*, and 4 patterns for *run-time fault prevention* (see Table 4). For refined design of these patterns, Grunske reviews the *watchdog*, *integrity check* and the *actuation monitor* (see also [18]) for failure/error detection. Similar to the scheme in [10], these patterns are described by the attributes *aliases, problem, context, forces, solution, rationales, resulting context*, and *related patterns.*

Component diagrams describe the transformation rules. Further aspects such as applying a behavioral model to these patterns are left open. However, the patterns are presented with sufficient detail to indicate the main ideas and their relationships. Although, these relationships are not described in detail. Particularly, the analysis of refinements or the relationship between run-time and design-time fault prevention (e.g. multi-channel redundancy with voting and HW platform substitution) would have been interesting. We consider it helpful to add the concept of requirement errors as a cause for systematic errors.

[18] Jari Rauhamäki, Timo Vepsäläinen, and Seppo Kuikka. "Functional safety system patterns". In: Proc. Nordic Conf. On Pattern Languages Of Programs (VikingPLoP); Tampere University of Technology, Department of Software Systems, Report 22*. 2012, pp. 48–68*

The authors discuss *HW and SW design patterns* for control and safety system development. They describe *separated safety* as their main pattern as well as *productive safety* (a pattern for high-level safe system behavior), *separated override*, *de-energized override*, *safety limiter* (a pattern for preventive safety actions), and *hardwired safety.* Each of the latter refines the *separated safety* pattern. The authors describe their patterns with respect to *context*, *problem*, *forces*, *solution*, *consequences*, *resulting context*, and *related patterns.*

Rauhamäki, Vepsäläinen, and Kuikka sketch structural and behavioral details of their patterns without using a specific method or language. Of particular interest is their *interdisciplinary discussion* of patterns, i.e., trying to capture reusable knowledge beyond the domain of software design. In this case, formal modeling could clarify some definitions (e.g. safety function, safe state), relationships between these patterns, as well as further details.

[14] Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. "Building a safety architecture pattern system". In: Proceedings of the 18th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Germany, July 10-14, 2013*. Ed. by Uwe van Heesch and Christian Kohls. ACM, 2013, p. 17.* ISBN*: 978-1-4503-3465-5*

The authors reflect on 15 widely known design patterns for fault-tolerant systems in safety-critical applications. Their approach includes the identification of the safety tactic (e.g. condition monitoring, degradation) underlying a pattern, construction of a GSN diagram to understand how the pattern implements this tactic, and the reconstruction of relationships (e.g. *similar*, *refines*) between these patterns. The patterns are listed in Table 4. Preschern, Kajtazovic, and Kreiner use a template with the attributes *name*, *type*, *also known as*, *context*, *problem*, *solution*, *GSN diagram*, *consequences*, *scenarios*, *uses*, and *credits*.

A *component model* describes the design underlying each pattern. *Graph models* represent the safety tactics. *GSN diagrams* convey arguments that a specific pattern implements a specific safety tactic and fulfills a corresponding subclass of safety goals. Based on comprehensible methodology, this pattern system forms a coherent contribution to the field. In this context, the authors could have described whether and how they qualified their classifications, e.g. by measuring inter-rater agreement.

[15] Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. "Security Analysis of Safety Patterns". In: Proc. 20th Conf. Pattern Languages of Programs*. PLoP '13. Monticello, Illinois: Hillside, 2013, 12:1–12:38*

This work extends [14] by security considerations. The authors discuss how security threats may occur for each of the 15 patterns. Therefore, they apply the STRIDE[2] approach, which is a structured way to identify threats using data flow diagrams. Preschern, Kajtazovic, and Kreiner complement the template used in [14] with the attribute *security GSN*, which contains a table of security flaws according to STRIDE. Similarly to [14], this approach uses GSN to establish a sufficient security argument. The authors explain their approach using a substation automation device case study.

This work integrates safety and security by taking into account, at a pattern level, how security threats could negatively influence safety properties of pattern applications.

5 Discussion and Conclusions

After having conducted the first stage of our study, we gained several insights into the field of safety patterns but also into our study design, which helps us

[2] The STRIDE approach encompasses the techniques of spoofing, tampering, repudiation, information disclosure, denial of services, and elevation of privilege.

Table 3. Characterization of Pattern Approaches.

	Engin.		Tech./Abs.			Language and Method	
Approach	Step	Cat.	SW	HW	M	Notation	Purpose
Hawkins and Kelly [5]	D	AP	✓	–	–	GSN+P	Software safety argumentation
Palin and Habli [12]	R	AP	✓	✓	✓	GSN+P+M	Safety case construction
Antonino and Trapp [1]	R	SP	✓	✓	–	UML, STR	Specify safety concepts consistent to architectural design
Antonino et al. [2]	R, D	PP	✓	✓	–	UML, FT	Safety requirements decomposition
Oertel, Kacimi, and Böde [11]	R, D, I	SP	✓	–	–	LTL-based STR, FT, SM	Safety specification vs. implementation compliance analysis
Grunske [4]	R, D	DP, TP	✓	✓	–	CD	Reliability design patterns for technical architecture transformations
Rauhamäki, Vepsäläinen, and Kuikka [18]	D	DP	✓	(✓)*	✓	CD, FFSD	Patterns for safety control applications
Preschern, Kajtazovic, and Kreiner [14]	D	DP, AP	✓	✓	–	GSN, CD	Safety goal argumentation
Preschern, Kajtazovic, and Kreiner [15]	D	DP, AP	✓	✓	–	GSN, DFD	Security goal argumentation

*The authors assume that the patterns can be extended to also consider electrical and electronic systems.

Legend: *Engineering step*: requirements management (R), design (D), implementation (I); *Category*: procedural pattern (PP), transformation pattern (TP), property specification pattern (SP), design pattern (DP), argumentation pattern (AP); *Technology/abstraction*: software (SW), hardware (HW), mechanics (M), system-level can be assumed if all three columns contain a checkmark; *Language and Method*: GSN with the extensions patterns and modules (GSN+P(+M)), unified modeling language (UML), structured textual requirements (STR), fault tree (FT), component diagram (CD), data-flow diagram (DFD), state machine (SM), flow of forces and substances diagram (FFSD)

improve the setup of the next two stages. Our findings and a critical reflection follows below.

5.1 Findings

As determined in Sect. 4, patterns are used at different *stages* during the system life cycle ranging from early requirements engineering and architectural design down to implementation. Thus, different *modeling methods and languages* are used fitting best the needs of safety engineers—mostly graphical notations like UML/SysML and GSN, but also textual patterns. Moreover, pattern attributes cover *software*, *hardware* (in the sense of electrical, electronic, and programmable electronic components), and mechanical aspects, such as in [18].

These insights helped us structure and characterize the selected pattern approaches. Their level of abstraction differs strongly. We observed that the explicit differentiation respectively the relationship between safety and *reliability* aspects of patterns is neglected in many of the selected publications, e.g. Preschern, Kajtazovic, and Kreiner [14] and Grunske [4]. Table 3 shows a characterization of

Table 4. Pattern Overview and Categorization

				Pattern Approaches								
Category	Pattern Identifier(s)	Principles (≤ 2, after [16])	Focused Technology	Hawkins/Kelly [5]	Palin/Habli [12]	Antonino/Trapp [1]	Antonino et al. [2]	Örtel et al. [11]	Grunske [4]	Rauhamäki et al. [18]	Preschern et al. [14]	Preschern et al. [15]
	high-level vehicle safety*		MX		✓							
	$\overset{sup}{\hookrightarrow}$ high-level SW safety*		SW	✓								
	$\overset{sup}{\hookrightarrow}$ SW contribution safety*		SW	✓								
	$\overset{sup}{\hookrightarrow}$ SW safety requirements*		SW	✓		(✓)	(✓)					
	$\overset{sup}{\hookrightarrow}$ argument justification SW*		SW	✓								
	$\overset{sup}{\hookrightarrow}$ through life safety				✓							
	predefined safety requirements				✓							
	$\overset{sup}{\hookrightarrow}$ homologation argument				✓							
AP	risk assessment				✓							
	risk management				✓							
	$\overset{sup}{\hookrightarrow}$ safety goal valid				✓							
	$\overset{sup}{\hookrightarrow}$ minimization				✓							
	$\overset{sup}{\hookrightarrow}$ alert and warning				✓							
	$\overset{sup}{\hookrightarrow}$ hazard. contrib. SW, risk mitig.*		MX	✓	✓	(✓)	(✓)					
	$\overset{sup}{\hookrightarrow}$ hazard identification		MX		✓							
	$\overset{sup}{\hookrightarrow}$ FMEA		MX		✓							
	product defects				✓							
	safety concept decomposition					✓	✓					
SP	param. safety con. spec. template					✓						
	safety contract							✓				
	HW platform reassignment	dt,Sub	HW						✓			
	HW platform substitution	dt,Sub	HW						✓			
	process fusion	dt,Sim	SW						✓			
	separated safety									✓		
	$\overset{gen}{\hookrightarrow}$ productive safety		MX							✓		
	$\overset{gen}{\hookrightarrow}$ hardwired safety		HW							✓		
	$\overset{gen}{\hookrightarrow}$ separated override	Ovr	HW							✓		
	$\overset{gen}{\hookrightarrow}$ de-energized override	Ovr	HW							✓		
	$\overset{gen}{\hookrightarrow}$ safety limiter	Msk	HW							✓		
	m-out-of-n, multi-ch. red. (w. voting)	rt,Rep,(Vot)	MX						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ triple modular red.		HW						✓		✓	✓
DP	$\overset{gen}{\hookrightarrow}$ m-out-of-n-d	Mon,Ovr	MX								✓	✓
	$\overset{gen}{\hookrightarrow}$ acceptance voting	San	SW								✓	✓
	$\overset{gen}{\hookrightarrow}$ homog. duplex, 2-ch. red.	rt	HW						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ heterog. duplex	Div	HW						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ recovery block	rt,Div,Rol	SW						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ n-version-programming	Div	SW								✓	✓
	$\overset{gen}{\hookrightarrow}$ n-self checking programming	Cmp	SW								✓	✓
	integrity check								✓			
	mon.-act., act. mon., sanity check	San,Mon	HW						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ watchdog	Hrt,Ovr	HW						✓		✓	✓
	$\overset{gen}{\hookrightarrow}$ 3-level safety monitoring		MX								✓	✓
	$\overset{gen}{\hookrightarrow}$ protected single channel	rt,Ovr	MX						✓		✓	✓
	safety executive	Deg,Ovr	HW								✓	✓
PP	Procedural pattern for safety requirements decomposition			✓	✓		✓					
TP	Design transformation rules for many DPs								✓			
AP'	GSN-based *safety* argumentation of DPs										✓	
AP"	GSN-based *security* argumentation of DPs											✓

*This pattern relationship was identified during our analysis. Pattern relationships identified by the other authors are not fully depicted in this table. Note that this table does not capture all of the relationships discussed in, e.g. [12, 14, 18]. Similar patterns are listed in the same row.

Legend: See Table 3. $A \overset{x}{\hookrightarrow} B$ denotes *A (sup)ported-by B, B contributes-to/in-context-of A* if $x = sup$ (AP), and *A (gen)eralizes B, B is-a/refines/specializes A* if $x = gen$ (DP). Run/design-time fault prevention (rt/dt), comparison (cmp), condition (mon)itoring, (deg)ration, (div)ersity, heartbeat (hrt), masking (msk), override (ovr), (rep)lication, (rol)lback, (san)ity check, (sim)plicity, (sub)stitution, (vot)ing, mixed (MX).

the reviewed pattern approaches, Table 4 gives an overview of the variety of discussed patterns.

5.2 Critical Reflection

A posteriori, we observed an accumulation of papers from two premier conference series for safety and patterns: SAFECOMP (international conference on computer safety, reliability & security) and PLoP (pattern languages of programs). This is not a problem per se but may indicate that for the next two study stages our search criteria need to be carefully reassessed, particularly, widened. Hence, the initially constructed search strings (cf. Table 1) potentially lead to a low coverage of the topic and research field.

It also turned out that search results only had an overlap of five publications between the two databases (Google Scholar and Elsevier Scopus). Search strings were too different in particular because the search engines did not allow semantically identical search queries. Consequently, we had to narrow down the Google Scholar results. Even with almost semantically similar queries, results strongly differed. Thus, we decided to treat search strings and results independently and manually merged relevant findings for the current work.

5.3 Conclusion

We presented first results of a three-staged study of *safety patterns*. For this, we performed a literature search in Google Scholar and Elsevier Scopus. The selected publications were reviewed to characterize this field of patterns, our main *goal* in the first stage.

One result of this first stage is the analysis of safety patterns of different categories in Table 4. An in-depth analysis of relationships between the patterns across these categories goes beyond the literature we studied so far. Furthermore, several insights help us to improve the field coverage, quality, and significance of the results of the follow-up stages. We need to ...

(i) carefully reassess search criteria to enhance research field coverage.
(ii) find an "optimal" search string per database, which fits best our reference keywords, rather than trying to find a common search string with weak expressive power. This, of course, includes synonyms for the term "pattern".
(iii) include relevant work found by additional manual search. This is particularly important with regard to the *mapping study* and the *systematic literature review.*
(iv) use exclusion criteria in addition to the ones listed in Sect. 3, e.g. citation count.
(v) find answers to research questions beyond the ones mentioned in Sect. 2, e.g. the extend to which safety, security, and reliability aspects of patterns are differentiated in the available approaches.

We perceive this paper, particularly, the questions posed in Sect. 2, as a basis we want to share in advance for a discussion about interesting research questions to answer as well as hypotheses to test in the next two stages of our study.

Acknowledgments

We would like to thank Maged Khalil for helpful feedback and discussions as well as our enthusiastic peer reviewers for their detailed comments making us effectively improve first versions of this study. Furthermore, we thank several safety engineers from BMW Group encouraging us to proceed with our study in the context of a third-party funded project.

Bibliography

Note: Annotated references are only listed inline.

[3] Hemangi Gawand, R.S. Mundada, and P. Swaminathan. "Design patterns to implement safety and fault tolerance". In: *International Journal of Computer Applications* 18.2 (2011), pp. 6–13. ISSN: 0975-8887.

[6] Gary G. Koch J. Richard Landis. "The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33.1 (1977), pp. 159–174. ISSN: 0006341X.

[7] T.P. Kelly. "A six-step method for the development of goal structures". In: *York Software Engineering, Flixborough, UK* (1997).

[8] Barbara Kitchenham. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Tech. rep. EBSE-2007-01. Ver. 2.3. School of CS, Math., Keele University, and Dept. of CS, U. Durham, 2007.

[9] Deborah Knott. *Writing an Annotated Bibliography*. online. University of Toronto. Dec. 2015. URL: http://www.writing.utoronto.ca/advice/specific-types-of-writing/annotated-bibliography.

[10] Gerard Meszaros and Jim Doble. "A Pattern Language for Pattern Writing". In: *Pattern Languages of Program Design*. Vol. 3. 1998, pp. 529–74. URL: http://hillside.net/index.php/a-pattern-language-for-pattern-writing.

[13] Kai Petersen et al. "Systematic Mapping Studies in Software Engineering". In: *12th Int. Conf. Evaluation and Assessment in Software Engineering (EASE), University of Bari, Italy, 26-27 June 2008*. Ed. by Giuseppe Visaggio et al. Workshops in Computing. BCS, 2008. URL: http://ewic.bcs.org/category/16334.

[16] Christopher Preschern, Nermin Kajtazovic, Christian Kreiner, et al. "Catalog of Safety Tactics in the light of the IEC 61508 Safety Lifecycle". In: *Proc VikingPLoP*. 2013, p. 79.

[17] Christopher Preschern et al. "Pattern-based safety development methods: overview and comparison". In: *Proc. 19th European Conf. Pattern Languages of Programs (EuroPLoP), Irsee, Germany, July 9-13, 2014*. Ed. by Veli-Pekka Eloranta and Uwe van Heesch. ACM, 2014, 28:1–28:20. ISBN: 978-1-4503-3416-7.