1 Motivation and Problem Statement

Industrie 4.0 promises that networked, rapidly reconfigurable automated factories can make manufacturing highly efficient [10]. However, established practices for manufacturing safety have little scope for reconfiguration: creating and validating a safety scheme takes a long time and a lot of highly specialized labor. That means that rapid reconfiguration would require decidedly un-rapid safety changes. To realize both functional safety and Industrie 4.0 rapid reconfiguration goals, it will be necessary to automate the creation and validation of safety measures.

Model-based Systems Engineering (MBSE) offers a set of tools which could solve this problem: it is possible to include both safety requirements and safety systems in a system model [18]. Theoretically, this would allow for automated verification of the safety conditions on the system. Extending this system model to a digital twin of a real-world system would thus allow for automated verification of safety conditions in a shorter amount of time. A further, implicit benefit here is the possibility of including safety alongside systems design, rather than as a second stage [26].

To begin work on this, researchers have developed conceptual frameworks for automating the functional safety process. Some have applied these frameworks in limited example facilities as well. However, at present, all have been forced to subjectively interpret relevant safety regulations in the contexts of their examples in order to create requirements that their system models can parse. In practice, this manual step is a severe bottleneck, preventing any practical gains from automation.

Etz et al. identify a first step towards bridging this gap: in addition to the configuration of the system model, it is necessary to have a knowledge base encoding knowledge about both the manufacturing system and the regulatory requirements, represented in a manner that allows for automated reasoning on the knowledge [14]. Following from this work, Reitgruber has developed a prototype knowledge base for manufacturing systems [29]. The other half remains open: developing a knowledge base for safety requirements. With this in place, it should be possible to apply automated reasoning to a system model and verify that it satisfies safety requirements.
2 Aim of the Work

In this work, we aim to develop a knowledge base for regulatory requirements, and to demonstrate its usefulness by modeling a real system and validating its safety characteristics using automated reasoning. We follow directly from Reitgruber’s work creating a knowledge representation for safety system components, turning to the question of representing and checking requirements.

We will structure our inquiry around the following research questions:

- System Model: How must existing system models be extended to enable their use in the evaluation of functional safety configurations?
- Rules and Checking: Which approaches can be employed to formulate functional safety rules and subsequently utilize them to automatically verify functional safety configurations?

3 State of the Art

We will first look at the technical topics in which our research questions are situated, and then take a more detailed look at current work in the field.

3.1 Technical Background

**Functional safety** is the part of overall safety focused on automatic changes in the behavior of a machine or system to prevent injury. One example is the control circuit that prevents an automatic door from closing on a person standing in the doorway [11]. This example is relatively passive: a more active example would be a controller on a wind turbine that angles the blades into a neutral position if wind speed increases above a safe limit. A number of standards guide functional safety: the most prominent is IEC 61508, which describes whole-lifecycle safety for electrical and electronic systems. A modification of this, IEC 62061, describes functional safety of safety-related control systems on machinery [27]. A range of other standards may also be relevant, depending on the industry and the jurisdiction. Alongside standards, governments also mandate industrial safety measures, to varying extents. In the European Union, the EU Machinery Directive (MD) sets the legal requirements for safety in manufacturing processes [3]. The relationship between IEC standards and the MD is complicated. For example, compliance with IEC 61508 does not give the presumption of compliance with the MD [7], but IEC 62061 does [2]. But regardless, in practice, standards and legal requirements are closely related, both in their development and their application. So, the IEC standards, together with the MD, provide an important and representative example of a regulatory framework for functional safety.

**Model-Based Systems Engineering** (MBSE) is the use of system models to support requirements, analysis, verification, and validation throughout a design process [5]. The essential goal of MBSE is to cope with the exponentially growing complexity of designed systems, in which component interactions, rather than component failures, become the primary causes of failure. The most common modeling tool for MBSE applications is SysML: while it is not necessarily a standard, it is widely used and required in some contexts, such as US DOD contracts [8]. In automation, AutomationML is another widely-used open standard [1]. There is increasing recognition that MBSE offers the possibility of solving safety engineering problems with which older safety methodologies are unable to cope. The question of how, exactly, to realize this is at the
forefront of systems engineering research today. For example, in her keynote at the INCOSE International Workshop 2023 MBSE Workshop, Nancy Leveson outlined the challenges of modern safety engineering and argued that advances in MBSE offered a way past them [6].

**Knowledge modeling** is the use of representations of real-world knowledge in structured frameworks to enable automated queries [22]. In the context of functional safety and MBSE, knowledge models are necessary in order to “ask” whether a system (model) is safe—or, equivalently, in order to state what, exactly, it means for a system to be safe. Knowledge models are commonly structured as ontologies: “set[s] of representational primitives with which to model a domain of knowledge or discourse” [16]. The most prevalent tool for describing ontologies is W3C’s OWL (“Web Ontology Language”) [31]. However, any given ontology itself is generally domain-specific. In this work, we will extend Reitgruber’s Reconfigurable Safety System Ontology, which is in turn developed from several domain-specific ontologies [29]. Attaching knowledge modeling to a MBSE workflow is the subject of current research in systems engineering [20] [9] [25] [4].

**Automated reasoning** is the use of software to check the truth of a statement [28]. In our context, we intend to form statements about the safety of our system model using our knowledge model, and then check the validity of such statements. While automated reasoning is, in full generality, an uncomputable problem, and is in virtually any practical application at least NP hard, practical approaches nonetheless exist. Most of these amount to reducing a given problem to a better-understood problem—boolean satisfiability is a common target—and then applying a well-developed solver for that problem—for example, Microsoft’s open-source Z3 solver, for satisfiability. These approaches are mature enough to solve such otherwise-intractable problems as software library dependency resolution in practical amounts of time. In particular, it is possible to use automated reasoning on knowledge models and system models, as our goals require [21].

### 3.2 Related Work

Research on integrating safety verification into MBSE (and, hence, into Industrie 4.0 rapidly reconfigurable systems) is still in early stages. We highlight some papers from the last several years showing development of frameworks for reaching this goal, and some offering limited demonstrations of MBSE approaches which point in the direction of this goal. We then dig deeper into recent criticism of pre-automation safety approaches, which serves to highlight the fundamental importance of automating safety validation in MBSE, even beyond Industrie 4.0 goals.

A number of researchers have produced conceptual frameworks for the project. Etz et al. propose a framework of five Service Groups for automated functional safety: our work fits into the first of these, Knowledge Representation [15]. Lee et al. highlight the importance of developing common ontologies and language for safety in systems engineering, with a focus on safety in process industry [23]. Salado presents several approaches to capturing requirements in model building—a necessary part of modeling safety requirements [30].

Alongside these theoretical development efforts, there are a number of implementation examples. Mhenni et al. demonstrate the possibility of introducing safety requirements analysis in the initial system design phase for an aircraft wheel brake subject to the ARP 4761 standard [26]. Hæring et al. demonstrate a full process of introducing safety requirements at the system engineering stage and checking their fulfillment with formal verification: in this case, their example is a malfunction indicator lamp for a car [18]. Javed et al. demonstrate the design of safety requirements for an autonomous guided vehicle using a design-by-contract approach [19]. And Bdwi et al. develop a set of safety requirements for an industrial robot and demonstrate the real-time checking of those requirements using camera-based safety systems in an experimental production unit [12].

In all cases above, however, the safety requirements are derived from the standards “manually”, by the engineers’ direct interpretation. Without knowledge modeling, interpreting safety standards in context
remains a critical bottleneck in the engineering process. Leveson highlights the essential problem with formulating safety requirements one-by-one: the complexity of designed systems is simply too high, and individual safety rules are unlikely to prevent emergent phenomena [21]. For example, the 1993 landing accident of a Lufthansa A320 in Warsaw was attributed in part to the design of the safety logic: the engine reversers and ground spoilers would not deploy unless both main landing gear detected enough load for the plane to be “on the ground” [32]. Due to other errors, the plane touched down with only one landing gear under load for nine seconds: in that time, commands to these braking features were blocked by the safety logic, and the plane overran the runway. Leveson presents this as an example of the limitations of component-focused safety design: while each component system performed according to its safety requirements, the emergent state was unsafe–fatally so for one crew member and one passenger. She argues for a “paradigm shift” from safety as reliability to safety as control: positively asserting the properties of globally safe states. This accords with Hollnagel’s much more philosophical treatment of safety nine years earlier: he argues that safety science must focus not on the unsafe states to be avoided, but on the characteristics of positively safe states [17].

The development frameworks mentioned above highlight the importance of knowledge modeling and automated reasoning for integrating safety into changeable system configurations. The example projects show, implicitly, the limitations on Industrie 4.0 design without these tools. And the demonstrated need to conceptualize global safety properties makes clear that reasoning on knowledge models will be necessary to connect the global level to the component level in a reliable way.

4 Methodology

To explore our research questions, we will follow a design science research methodology. We adopt the six-step cycle described by vom Brocke et al. in “Introduction to Design Science Research” [13].

1. Problem identification and motivation: We refine the above research question into a clear problem statement about checking safety requirements using automated reasoning.

2. Define the objectives for a solution: Our objectives will relate to the four major choices we have to make. We will develop the basic objectives below as we complete the literature review:

   • Example use case: Our use case should relate to a specific industrial process, and should admit multiple useful configurations. It should be possible to realize a physical or simulated model of the use case, and to rapidly reconfigure this.
   • Example safety requirement(s): The safety requirement(s) chosen should come from real standards related to the use case. They should be simple enough to admit essentially complete semantic description and use in automated reasoning.
   • Knowledge representation format: The chosen knowledge representation format should be well-suited to describing both industrial systems and regulatory requirements. It should be amenable to automated reasoning.
   • Automated reasoning solution: The chosen automated reasoning tool(s) should be fast enough to check configurations in seconds or minutes. The tool(s) should be freely available for use.

3. Design and development: We will build an example use case context and a tool for checking the safety compliance of configurations in that context.

4. Demonstration: We will demonstrate our model solution on multiple configurations of the model use case.
5. Evaluation: Based on the objectives we have created, we will evaluate our tool. We will evaluate our knowledge representation qualitatively, and our automated reasoning solution quantitatively (e.g., execution time).

6. Communication: We will present our work in our thesis paper in which we integrate our results into the framework proposed by Etz et al. [14].

5 Curriculum Relevance

Within the Computer Engineering curriculum, the Automation and Computer-Aided Verification modules are most closely relevant to this work. The most relevant courses are:

- 185.291/A93 Formal Methods in Computer Science
- 181.144/145 Computer-Aided Verification
• 191.104 Information Technology in Automation
References


