Sarah Moniky Silva Ribeiro

INTEGRATING REQUIREMENTS MODELING AND SAFETY ANALYSIS: THE CASE OF ISTAR4SAFETY AND STPA

Ph.D. Thesis

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2025

Federal University of Pernambuco
Center for Informatics
Graduate in Computer Science

Sarah Moniky Silva Ribeiro

INTEGRATING REQUIREMENTS MODELING AND SAFETY ANALYSIS: THE CASE OF ISTAR4SAFETY AND STPA

A Ph.D. Thesis presented to the Center for Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science.

Main Area: Software Engineering and Programming Languages

Advisor: Jaelson Freire Brelaz de Castro
Co-Advisor: Ricardo Argenton Ramos

RECIFE
2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Ribeiro, Sarah Moniky Silva.
Integrating Requirements Modeling and Safety Analysis: The Case of iStar4Safety and STPA / Sarah Moniky Silva Ribeiro. - Recife, 2025.
236f.: il.

Tese (Doutorado)- Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-graduação Acadêmico em Ciência da Computação, 2025.
Orientação: Jaelson Freire Brelaz de Castro.
Coorientação: Ricardo Argenton Ramos.

1. Sistemas críticos de segurança; 2. iStar4Safety; 3. GORE; 4. STPA; 5. RESafety; 6. Engenharia de requisitos. I. Castro, Jaelson Freire Brelaz de. II. Ramos, Ricardo Argenton. III. Título.

UFPE-Biblioteca Central

Sarah Moniky Silva Ribeiro

“Integrating requirements modeling and safety analysis: the case of iStar4Safety and STPA”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Engenharia de Software e Linguagens de Programação.

Aprovada em: 28/08/2025.

_____

Orientador: Prof. Dr. Jaelson Freire Brelaz de Castro

BANCA EXAMINADORA

_____

Profa. Dra. Carla Taciana Lima Lourenço Silva Schuenemann
Centro de Informática / UFPE

_____

Prof. Dr.Johnny Cardoso Marques
Divisão de Ciência da Computação/ITA

_____

Prof. Dr. Luiz Eduardo Galvão Martins
Departamento de Ciência e Tecnologia/UNIFESP

_____

Prof. Dr. Renato de Freitas Bulcão Neto
Instituto de Informática/UFG

_____

Profa. Dra. Fernanda Maria Ribeiro de Alencar
Departamento de Eletrônica e Sistemas/UFPE

I dedicate this thesis to the memory of Karen Emanuelly.

Acknowledgements

(Although this thesis is written in English, the acknowledgments remain in Portuguese.)

Esta tese é fruto de muito esforço e de anos marcados por acontecimentos paralelos que, por pouco, não impediram sua conclusão. Finalizar um doutorado é, acima de tudo, um ato de persistência e resiliência. Durante essa trajetória, aprendi imensamente, amadureci como pessoa e como pesquisadora, e levei comigo lições que permanecerão para toda a vida.

Agradeço, primeiramente a minha mãe, Maria José, e a meu pai, Zilmar, pelo amor e pelo cuidado dedicados a mim. Nunca serei capaz de retribuir tanto. Obrigada, Sarah, sempre comigo e torcendo por cada uma das minhas conquistas. Obrigada aos meus sobrinhos, Maria Fernanda, Alice, Lorenzo, Anthony e André, luzes constantes na minha vida. Obrigada, Ana Luiza. E, especialmente, como já expresso na epígrafe, dedico este trabalho a Karen, minha irmã que nos deixou, e que permanece viva em nossa memória e em nossos corações.

Agradeço ao meu orientador, professor Jaelson Castro, que me acompanha desde o mestrado, orientando-me não apenas academicamente, mas também sendo um apoio em momentos pessoais, sempre com paciência e incentivo. Agradeço também ao meu coorientador, professor Ricardo, por aceitar embarcar nesta jornada, pelas sugestões valiosas e pelos feedbacks que enriqueceram este trabalho.

Aos meus amigos, que de tantas formas foram meu suporte, meu afeto e meu alento, Aos amigos de Montes Claros, de Recife e da Lapa, amo vocês. Sem vocês, essa trajetória seria imensamente mais pesada.

Aos professores membros da banca, agradeço a disponibilidade em ler esta tese, participar desta avaliação e contribuir com seus preciosos comentários e sugestões para aprimorá-la.

Agradeço à FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco) pelo apoio financeiro, e ao CIn/UFPE por ter sido meu lar acadêmico, permitindo-me trilhar minha jornada de pós-graduação em terras pernambucanas. Agradeço também ao IFBaiano-Campus Lapa, instiuição onde hoje atuo como professora, por me dar condições de terminar esse trabalho.

Por fim, deixo minha saudação a todos que acreditam na ciência e na pesquisa, e que trabalham para que estas continuem avançando com qualidade. Que nunca nos faltem forças para combater a desinformação e coragem para seguir buscando resultados sólidos e relevantes.

Resumo

Sistemas Críticos de Segurança (SCSs) são inerentemente complexos e não triviais, compostos por numerosos componentes interdependentes cujas interações podem levar a comportamentos perigosos não intencionais. Tratar adequadamente sua segurança exige uma análise rigorosa e sistemática desde os estágios iniciais de desenvolvimento, considerando que muitas falhas relacionadas à segurança têm origem na fase de engenharia de requisitos. Embora a Engenharia de Requisitos (ER) e a Engenharia de Segurança (ES) compartilhem preocupações complementares, a integração entre ambas ainda carece de avanços. Em particular, observa-se uma lacuna quanto ao suporte sistemático para derivar requisitos de segurança a partir da análise de perigos e para representar, em modelos de requisitos compreensíveis e rastreáveis, os elementos de segurança identificados nessas análises. A ausência dessa integração pode resultar em requisitos de segurança que não se alinham ao desenvolvimento global do sistema, na omissão de requisitos relevantes ou em sua definição de forma ambígua ou inconsistente. Além disso, pode comprometer a rastreabilidade entre elementos não relacionados à segurança e elementos de segurança, dificultando a verificação e a validação do sistema como um todo. Para enfrentar esse desafio, propomos o RESafety, um processo destinado a apoiar a modelagem inicial de requisitos de segurança, integrando modelos de objetivos de uma linguagem de modelagem orientada a objetivos com a análise produzida por uma técnica de segurança. O RESafety é estruturado em sete etapas iterativas, representadas por meio de BPMN para favorecer a compreensão e o rastreamento. O processo foi aplicado a dois sistemas críticos, um Sistema de Bomba de Infusão de Insulina e um Sistema de entrega de medicamentos por um braço robótico no contexto da farmácia de um hospital. Sua avaliação empírica foi conduzida com especialistas em Engenharia de Requisitos e Engenharia de Segurança, utilizando o Modelo de Aceitação de Tecnologia (TAM). Os resultados indicam uma percepção predominantemente positiva quanto à utilidade e à facilidade de uso do RESafety, embora tenham sido apontadas limitações e sugestões de melhoria, que foram categorizadas e analisadas.

Palavras-chave: Sistemas Críticos de Segurança, iStar4Safety, GORE, STPA, RESafety, Engenharia de Requisitos, Análise de Segurança, Análise de Perigos.

Abstract

Safety-Critical Systems (SCSs) are inherently complex and non-trivial, composed of numerous interdependent components whose interactions may lead to unintended hazardous behaviors. Adequately addressing their safety requires a rigorous and systematic analysis from the early stages of development, considering that many safety-related failures originate during the requirements engineering phase. Although Requirements Engineering (RE) and Safety Engineering (SE) share complementary concerns, their integration still lacks significant progress. In particular, there is a gap regarding systematic support for deriving safety requirements from hazard analysis and for representing, in understandable and traceable requirements models, the safety elements identified in such analyses. The absence of this integration may result in safety requirements that are misaligned with the overall system development, in the omission of relevant requirements, or in their definition in ambiguous or inconsistent ways. Furthermore, it may compromise the traceability between non-safety and safety-related elements, hindering the verification and validation of the system as a whole. To address this challenge, we propose RESafety, a process designed to support the early modeling of safety requirements by integrating goal models from a goal-oriented modeling language with the analysis produced by a safety technique. RESafety is structured into seven iterative steps, represented through BPMN to enhance understanding and traceability. The process was applied to two critical systems: an Insulin Infusion Pump System and a Medication Delivery System based on a robotic arm operating within a hospital pharmacy context. Its empirical evaluation was conducted with experts in Requirements Engineering and Safety Engineering, using the Technology Acceptance Model (TAM). The results indicate a predominantly positive perception regarding the usefulness and ease of use of RESafety, although some limitations and suggestions for improvement were identified, categorized, and analyzed.

Keywords: Safety-Critical Systems, iStar4Safety, GORE, STPA, RESafety, Requirements Engineering, Safety Analysis, Hazard Analysis.

List of Figures

List of Tables

Acronyms

BPMN	Business Process Model and Notation	22
ETA	Event Tree Analysis	17
FMEA	Failure Modes and Effects Analysis	17
FTA	Fault Tree Analysis	17
GORE	Goal-Oriented Requirements Engineering	16
HAZOP	Hazard and Operability Analysis	17
HC	Hazardous Conditions	80
IIPS	Insulin Infusion Pump System	16
KAOS	Keep All Objectives Satisfied	19
LS	Loss Scenario	84
PEOU	Perceived Ease of Use	23
PSA	Preliminary Safety Analysis	19
PU	Perceived Usefulness	23
RE	Requirements Engineering	16
SCS	Safety-Critical Systems	16
SD	Strategic Dependency	69
SR	Strategic Rationale (iStar model) / Safety Requirement (RESafety element)	69
STAMP	System-Theoretic Accident Model and Processes	18
STPA	System-Theoretic Process Analysis	18
TAM	Technology Acceptance Model	22
UCA	Unsafe Control Action	80
SC	Safety Constraints	72
R	Responsabilities	72
H	System-Level Hazards	73
A	Accidents	73
RE	Requirements Engineering	16
SE	Safety Engineering	17

Contents

1 Introduction

This chapter presents the context and motivations that led to the development of this research. It defines the main research questions and specific objectives, explains the methodology adopted, and provides a summary of the publications produced during the course of the work. Finally, it outlines the structure of the thesis.

1.1 Context

A system can be understood as a union of interacting components—including software, hardware, people, processes, and data—that operate in complex and interdependent ways, with intricate dependencies among their elements. These systems promote the intersection of two key areas that are central to this work: Safety Engineering and Requirements Engineering (Leveson, 1995; Scholz & Thramboulidis, 2013). Such systems may perform basic, critical, or automated functions. They may also be considered, due to their critical nature, as Safety-Critical Systemss (SCSs), such as Insulin Infusion Pump System (IIPS), Air Traffic Control Systems, and socially assistive robots(Knight, 2002; Martins *et al.*, 2015; Sharifi *et al.*, 2020; Albuquerque *et al.*, 2017).

The development process of such systems involves several phases. In complex and safety-critical contexts, activities such as feasibility analysis, concept of operations (ConOps), and preliminary hazard analysis typically precede the Requirements Engineering (RE). However, once the requirements engineering phase is reached—one of the earliest in the development cycle—requirements are elicited, analyzed, and defined to establish a solid foundation for subsequent activities. This stage is particularly critical because errors or omissions in requirements can propagate to later phases, leading to increased costs, project delays, and compromised system quality and safety.

In the early stages of RE, the use of *goals* has proven effective for organizing and justifying requirements (Houhamdi *et al.*, 2024; Alturayeif & Hassine, 2025). Goal-Oriented Requirements Engineering (GORE) languages support a more holistic and precise definition of requirements by modeling them from a high-level perspective. These languages emphasize pre-

requirement activities, representing system goals and their responsible actors Yu (1997); Vilela *et al.* (2017b). The models or diagrams produced by such approaches address the need to better understand complex phenomena by abstracting them into more comprehensible representations. These artifacts play a crucial role in facilitating communication between analysts and developers, while also offering a holistic view of the system to be built(Leveson, 1995).

Despite its advantages, the use of Goal-Oriented Requirements Engineering (GORE) introduces specific challenges when compared to traditional approaches. Analysts and stakeholders often require specialized training to understand and apply goal-oriented notations such as i* or KAOS, which can limit the adoption of these methods in industrial contexts. The integration of goal models with other engineering artifacts, such as behavioral or architectural models, also presents difficulties, as maintaining consistency and traceability across heterogeneous representations demands additional effort and tool support. Furthermore, ensuring the scalability and maintainability of goal models throughout iterative development cycles remains a significant challenge, since continuous updates are needed to preserve coherence with evolving system objectives.

In this thesis, the focus is on the safety analysis of a specific class of systems known as SCSs. These are systems in which the occurrence of certain hazards may lead to accidents with severe consequences, including human injury or death, economic loss, environmental damage, or mission failure (Leveson, 2011, 1995).

During the life cycle of a SCS, methods, and processes must be used to deal with the complexity of their development, seeking to develop a safer system both at the level of components (or subsystems) and at the level of a complex interaction between them. Safety must be considered from the beginning of the SCS development process until the end of its useful life Martins & Gorschek (2021); Leveson (1995).

In the field of Safety Engineering (SE), traditional safety analysis techniques have long been used to identify potential hazards—even before the emergence of computerized systems, in which software has become an indispensable component. However, these traditional approaches typically emphasize the reliability of individual components and fail to consider safety as an emergent property of the system. Moreover, they do not adequately address computational control or the increasing complexity of modern technologies. Firesmith (2004) classifies such techniques—including Event Tree Analysis (ETA), Fault Tree Analysis (FTA), and Failure Modes and Effects Analysis (FMEA)—as reliability-focused. Although Hazard and Operability Analysis (HAZOP) is recognized as a safety analysis method, it similarly adopts a reductionist perspective centered on component-level evaluation. This limitation is problematic because safety in modern systems often emerges from the interactions between components rather than from their individual reliability. As a result, relying solely on component-level analysis may overlook critical hazards. Therefore, it is necessary to adopt approaches that treat safety as an emergent system property, capable of capturing control dynamics, interdependencies, and the growing complexity of technological systems.

In contrast, System-Theoretic Process Analysis (STPA), derived from the System-Theoretic Accident Model and Processes (STAMP) framework proposed by Leveson (2011), represents a significant advancement. It conceptualizes safety as an emergent property that arises from the dynamic interactions among system components. This systems-theoretic approach offers a broader and more integrated perspective than its predecessors, making it particularly suited to analyzing the complexity of modern socio-technical systems, which are systems that integrate human, organizational, and technical entities (Sheikh Bahaei & Gallina, 2024).

RE often faces challenges in adequately addressing safety concerns, making it necessary to involve safety engineers to perform safety analysis (Leveson & Thomas, 2018; Knight, 2002). Conversely, safety engineers require appropriate techniques to model the various safety requirements identified throughout their analyses. Given that natural language is inherently ambiguous, there is a need for more precise modeling languages to represent these requirements accurately and support effective communication across disciplines.

Moreover, establishing a common language between professionals in the fields of RE and SE remains a significant challenge. Defining a shared terminology in any domain is inherently difficult, yet critically important, as it facilitates communication among stakeholders—a factor that is essential to the success of any project (Leveson, 1995; Vilela *et al.*, 2017b).

Considering all the points discussed, the objective is to bridge the gap between the two domains—RE and SE. On one hand, a GORE approach is adopted, as it provides high-level and socially-aware visualizations of system components, enabling rationale traceability and the representation of safety concerns. On the other hand, a hazard analysis technique is incorporated which, although not specifically designed to identify safety requirements, can effectively support this process by maintaining a focus on both system components and their complex interactions.

To achieve this goal, RESafety is proposed as a six-step process designed to support the analysis and modeling of safety requirements from the early stages of developing a SCS. RESafety is grounded in iStar4Safety (Ribeiro *et al.*, 2019a), an extension of the GORE language iStar 2.0, and in STPA (Leveson, 2011), a hazard analysis technique based on the STAMP framework.

The following section outlines the main problems in the current state of the field and the motivations underpinning this research.

1.2 Problem and Motivation

Accidents in safety-critical domains can lead to severe financial, environmental, and human losses. A well-documented factor contributing to system failures is faulty or missing requirements (Leveson & Thomas, 2018; Martins & Gorschek, 2021; Berry, 1998). Consequently, RE represents one of the earliest and most critical phases in the development of any system, particularly SCS.

Safety concerns must be addressed from the very beginning of the software development

process and maintained throughout the system's lifecycle—covering both development and documentation phases—to prevent accidents (Leveson, 2011). This implies that safety requirements must be as complete, correct, and up to date as possible. However, defining requirements specifications that are complete, unambiguous, testable, and understandable remains a major challenge for organizations developing SCSs (Martins & Gorschek, 2017). This scenario highlights the need for approaches that support better modeling of safety requirements, enabling documentation that can be continuously updated and maintained.

Top-down system engineering has been recognized as essential for integrating safety into complex systems (Leveson, 2011). This reinforces the need for iterative and incremental refinement of requirements from the early phases of RE. GORE has proven effective in supporting this process, as it provides a structured means to organize and justify software requirements (Anton, 1996). Furthermore, many GORE languages enable the modeling of social relationships and interactions among system actors, allowing the analysis of interdependencies rather than focusing solely on isolated components. In addition, models in general are powerful tools for understanding complex phenomena and for communicating them to stakeholders in a more comprehensible way (Leveson, 1995). Hence, addressing safety from the earliest stages of system development is critical, with GORE providing a structured and effective means to support this endeavor.

In current practice, early safety requirements are typically captured in a Preliminary Safety Analysis (PSA) document, which serves as the foundation for subsequent analyses (Leveson, 1995; Leveson & Thomas, 2018). Prior research (Vilela *et al.*, 2017b) has identified key features that requirements languages should provide to support early safety requirements specification and has highlighted GORE languages like iStar and Keep All Objectives Satisfied (KAOS) as promising candidates. Nevertheless, these languages lack the expressiveness needed to capture all relevant safety elements (Vilela *et al.*, 2017b). This limitation motivated the development of iStar4Safety (Ribeiro *et al.*, 2019b), an extension of the iStar 2.0 language specifically tailored to model essential elements of early safety requirements. Its systematic creation, encompassing key elements for safety analysis, along with empirical studies that evaluated its acceptability for modeling safety requirements in previous works, supports its suitability in this context. Combined with the lack of other GORE languages capable of adequately representing safety aspects, these factors justify the selection of iStar4Safety as the integrating GORE language for the process proposed in this thesis.

Moreover, accidents cannot be adequately understood solely by examining event chains or individual component failures. Evidence shows that accidents in modern systems often stem from complex interactions among system components, making safety an emergent property rather than a characteristic of isolated parts. Traditional safety analysis techniques based on scientific reductionism are thus insufficient for handling contemporary complex systems. A system-theoretic approach is required—one that considers safety as a control problem and analyzes the interdependencies between components. STPA has emerged as an innovative

technique in this regard, offering a structured approach to model unsafe control actions and their causal factors (Leveson, 1995, 2011).

Despite the need to address safety from the early stages of system development, a significant gap persists between RE and SE practices. The core problem lies in the lack of integration between these two areas: while RE focuses on systematically capturing and modeling stakeholder goals and system requirements, SE emphasizes hazard identification and safety analysis. Without a common modeling language or a process to unify them, communication among stakeholders becomes fragmented, leading to safety requirements that are incomplete, ambiguous, or not aligned with overall system development (Leveson, 1995; Broomfield & Chung, 1997; Vilela *et al.*, 2017b,a). While efforts such as the taxonomy proposed by Vilela *et al.* (2017a) aim to establish a common background for RE and SE, there is still a need for processes that effectively integrate these areas, particularly by combining early-phase GORE modeling with system-theoretic safety analysis techniques.

This research is motivated by the need to bridge this gap. To this end, RESafety is proposed as a process that integrates iStar4Safety, to model early safety requirements from the RE perspective, with STPA, to perform system-theoretic safety analysis. The definition of its steps is intended to enable the modeling of relevant safety elements identified during STPA-based hazard analysis, thereby fostering improved communication and alignment between the RE and SE domains and contributing to the development of safer systems.

1.3 Research Questions and Objectives

Based on the context and motivations presented, this thesis first defines its core research problem, which guided the conception and development of this work:

> *How can Requirements Engineering and Safety Engineering be systematically integrated—through a goal-oriented approach and a system-theoretic safety analysis technique—to support the early modeling and traceability of safety requirements in safety-critical systems?*

From this problem, the following Research Questions (RQ) were derived to guide the investigation and structure the methodological approach of this thesis:

RQ01 ***Which Goal-Oriented Requirements Engineering (GORE) language and safety analysis technique are suitable for supporting the modeling of safety requirements in safety-critical systems?***

The first research question is motivated by the need to identify appropriate methods that can effectively capture and reason about safety requirements from the early stages of system development. While several GORE languages exist, their suitability for safety-critical domains is not well established. Similarly, safety analysis techniques vary in scope and

rigor, and not all of them are equally effective for integration with requirements modeling. Therefore, it is essential to investigate which combination of a GORE language and a safety analysis technique provides the most adequate support for modeling safety requirements in safety-critical systems.

RQ02 ***How can elements of Goal-Oriented Requirements Engineering models be systematically interrelated with those from safety analysis techniques to support integrated safety modeling?***

The second research question emerges from the gap between Requirements Engineering and Safety Engineering practices. Based on the analysis of the techniques identified as suitable for integration, there is still a lack of systematic approaches to link GORE models—focused on stakeholders' goals, dependencies, and rationales—with safety analysis models, which may produce hazards, accidents, and mitigation strategies. Without such integration, safety concerns may remain disconnected from early requirements models, resulting in omissions or inconsistencies. Thus, this question aims to explore how to establish systematic and traceable interrelations between GORE and safety analysis elements to support a coherent and integrated modeling process.

RQ03 ***What are the perceived usefulness, perceived ease of use, and expert-suggested improvements regarding the RESafety process?***

The third research question is motivated by the need to empirically evaluate the proposed RESafety process with practitioners and experts. Beyond its theoretical design, the practical value of RESafety depends on how users perceive its ability to support the elicitation, modeling, and traceability of safety requirements. By assessing perceived usefulness and ease of use—alongside collecting expert feedback and improvement suggestions—this question seeks to demonstrate that the proposed process is not only methodologically sound but also acceptable and applicable in real-world safety-critical system development contexts.

Our objective is to propose a process for integrating Requirements Engineering (RE) and Safety Engineering (SE) by combining a Goal-Oriented Requirements Engineering (GORE) modeling technique with a system-theoretic safety analysis approach. This integration aims to enable the systematic modeling of safety requirements derived directly from hazard analysis.

To address this general objective, this thesis defines the following specific objectives:

O1 - Identify and compare existing approaches in both GORE and safety analysis that are appropriate for use in SCS contexts. This includes determining suitable techniques for integration.

- Review existing safety analysis techniques and select a suitable one.
- Review existing GORE languages that support safety modeling and select a suitable one.

O2 - Define a process to support the early modeling of safety requirements by integrating goal models with a safety analysis technique. It should support the systematic modeling of safety requirements derived from the hazard analysis.

- Understand how the elements of both approaches are related.
- Identify necessary adjustments to enable the combined use of both approaches.
- Define an approach called RESafety, the steps, activities, and artifacts of the proposed process, and represent it using the Business Process Model and Notation (BPMN).
- Provide illustrations of the use of the RESafety Process.

O3 - Conduct an empirical evaluation of the proposed RESafety process with the participation of Requirements and Safety experts. The objective is to assess the process's usefulness and ease of use in practice, from the perspective of professionals in RE and SE. Additionally, this evaluation aims to gather constructive feedback and identify potential areas for improvement, using the Technology Acceptance Model (TAM) as the guiding evaluation framework. Specific sub-objetives:

- Design a survey instrument to enable a static evaluation of the process from the perspective of requirements and safety engineers.
- Analyze the feedback and apply necessary improvements and corrections to the process.

The purpose of **Specific Objective O1** is to explore the current landscape of GORE languages and safety analysis techniques suitable for SCS development. This objective involves reviewing, comparing, and selecting appropriate approaches from both domains, based on their expressive capabilities, methodological rigor, and practical applicability. By identifying techniques that can be effectively integrated, this objective lays the foundation for supporting the early modeling of safety requirements using a combined GORE and safety analysis perspective.

Specific Objective O2 aims to investigate how the selected GORE language—specifically its safety-oriented constructs—and the chosen safety analysis technique can be methodologically aligned. This includes understanding how the modeling elements and artifacts from both approaches relate to each other and identifying the necessary adaptations to enable a coherent and traceable integration. Moreover, this objective involves defining the steps, activities, and artifacts that compose the RESafety process, supported by BPMN representations. The process is then

applied to both an example from the literature and a real-world SCS to illustrate its applicability and to refine it based on practical modeling experience.

The purpose of **Specific Objective O3** is to conduct an empirical evaluation of the RESafety process through the participation of experts in RE and SE. It seeks to assess the practical effectiveness and usability of the process, identifying strengths and limitations based on expert feedback. The evaluation is grounded in the TAM, focusing on the constructs of Perceived Usefulness (PU), Perceived Ease of Use (PEOU), and suggested improvements. The collected insights will be used to further refine the process, thus contributing to answering.

1.4 Research Methodology

This section outlines the methodological foundation adopted in this thesis. The choice of an appropriate research methodology is crucial to ensure the rigor and reliability of the results, as it provides a structured path to address the research questions and achieve the proposed objectives. Below, the classification of this research is presented, together with the steps undertaken for its development.

1.4.1 Research Classification

In Easterbrook *et al.* (2008), the authors present key questions to support the selection of an appropriate research method, ranging from philosophical considerations to practical aspects related to its application. Following these guidelines, the research was classified within a suitable methodological group. Considering the aspects discussed and the classification proposed by Easterbrook *et al.* (2008), Table 1.1 positions the research activities of this thesis within the corresponding methodological framework.

Table 1.1: Methodological framework

Aspect	Classification
Types of Research Question	Exploratory: Description and Classification, Design
Philosophical stance	Pragmatism
Nature	Qualitative and Quantitative
Research methods	Illustrative scenario, and qualitative survey with experts

Source: The author (2025)

1.4.2 Research Steps

This work is developed through the empirical and engineering methods presented by Glass (1994). The engineering method involves the observation of existing solutions, proposing improvements or developing new ones through iterative measurement and analysis, until no

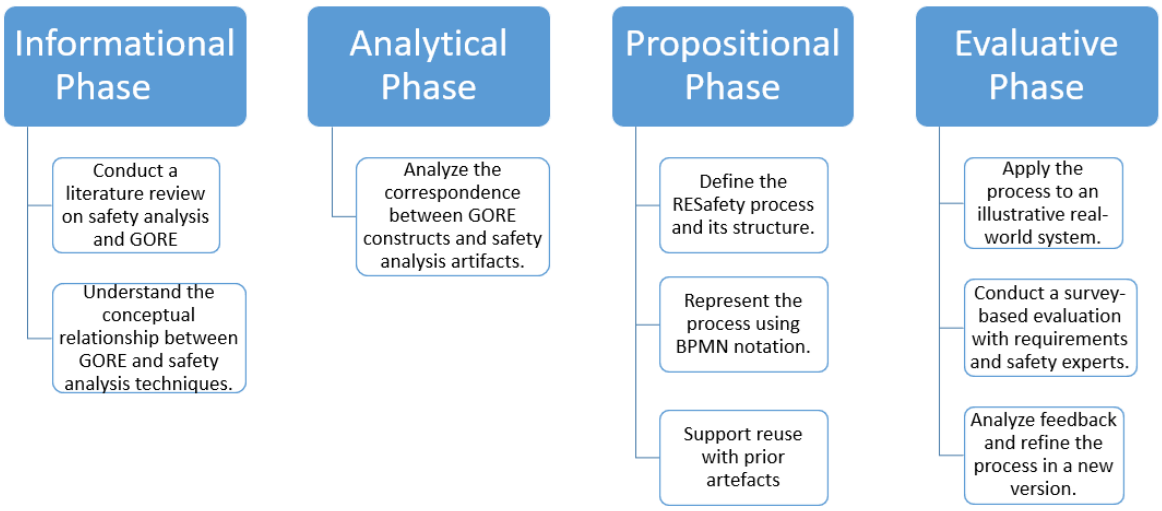
further significant enhancements can be identified. The empirical method, in turn, focuses on evaluating the proposed solution through use cases, experiments, or surveys (Glass, 1994; Wohlin *et al.*, 2012).

To support the development of this research, the following phases of the engineering method proposed by Glass (1994) are adopted: the informative phase, the analytical phase, the propositional phase, and finally, the evaluative phase. These phases are detailed below:

- **Informational phase**: This phase focused on gathering and consolidating knowledge through literature review and conceptual understanding. In this work, a study was conducted to review safety analysis techniques and GORE, aiming to understand their conceptual relationship.
- **Analytical phase**: This phase involved examining the correspondence between GORE constructs and safety analysis artifacts. In the present research, this included analyzing how elements of iStar4Safety aligned with components of safety techniques such as STPA, identifying overlaps and gaps that informed the design of the proposed approach.
- **Propositional phase**: The propositional phase aimed to define the RESafety process and its internal structure. In this phase, the process was represented using BPMN, and strategies were proposed to support the reuse of previously generated artifacts and models, consolidating the foundations of the proposed approach.
- **Evaluative phase**: In this final phase, the proposed process was applied to an illustrative real-world system to validate its feasibility. A survey-based evaluation was conducted with safety and requirements experts, allowing feedback collection and refinement of the process for a future version.

Thus, Figure 1.1 presents an overview of the activities carried out in each phase of the engineering method during this research.

Figure 1.1: Phases of the engineering method and corresponding activities in this work.



Source: Author (2025).

1.5 Summary of Publications

This section presents the publications and the book chapter resulting from the research conducted in this thesis.

1. RIBEIRO, M.; CASTRO, J. Modeling Early Safety Requirements with Requirements4Safety. In: *Proceedings of the 25th Workshop on Requirements Engineering (WER 2022)*, 2022, Natal (Virtual Mode). Published in the Masters and Doctoral Track (WER-MDT), 2022. Originally published in Portuguese. pp. 1–10.

2. RIBEIRO, M.; CASTRO, J.; RAMOS, R.; LENCASTRE, M.; SANTOS, A. Requirements4Safety: Developing a Technique for Modeling Initial Safety Requirements. (in Portuguese Requirements4Safety – Construindo uma Técnica para Modelagem de Requisitos Iniciais de Segurança). In: *Proceedings of the Workshop on Requirements Engineering (WER 2023)*, 2023, Porto Alegre. Originally published in Portuguese.

3. RIBEIRO, M.; CASTRO, J.; ARGENTON, R.; LENCASTRE, M.; SANTOS, A.; PASTOR, O. Integrating Goal-Oriented Requirements Modeling and Safety Analysis with Requirements4Safety. In: *Proceedings of the 16th International iStar Workshop (iStar 2023) co-located with the 31st IEEE International Requirements Engineering Conference (RE 2023)*, 2023, Hannover, Germany. vol. 16, pp. 40–46.

4. RIBEIRO, M.; CASTRO, J.; ARGENTON, R. Integrating STPA with Safety Requirements Modeling. In: *Proceedings of the 38th Brazilian Symposium on Software Engineering (SBES 2024)*, 2024, Brazil. p. 561-567.

5. RIBEIRO, M.; CASTRO, J.; LENCASTRE, M. Goal-Oriented Modeling of Safety-Critical Systems. In: FRANCH, X.; LEITE, J. C. S. P.; MUSSBACHER, G.; MYLOPOULOS, J.; PERINI, A. (Eds.). *Social Modeling Using the i* Framework*. 1st ed. Cham, Switzerland: Springer, 2024. vol. 18, pp. 101–117.

1.6 Thesis Structure

This thesis is organized into six chapters.

Chapter 1 presents the context of the work, the motivation and rationale behind its development, the research questions and objectives that guided its execution, the research methodology adopted, and a summary of the publications produced during its development.

Chapter 2 introduces the theoretical foundations related to GORE, SCS, and safety analysis, including descriptions of the STPA technique, the iStar4Safety language, and the TAM model. In addition, related work that supports and complements this thesis is discussed.

Chapter 3 describes the RESafety process, detailing each of its six modeling steps and illustrating the approach through the modeling of an IIPS. Furthermore, alternative proposals for applying RESafety are also presented.

Chapter 4 presents the application of the RESafety process to a real SCS, demonstrating each modeling step, the generation of safety analysis artifacts, and the resulting final safety assessment.

Chapter 5 presents the evaluation of the RESafety process through an empirical study conducted with experts in RE and SE, using a TAM-based survey instrument.

Chapter 6 concludes the thesis by summarizing the main findings, answering the research questions, and presenting the contributions, limitations, and directions for future work.

In the following chapter, the background and related works that support this research are presented.

1.7 Chapter Summary

This chapter provided an introduction to the thesis. It presented the context that motivated its development, followed by the underlying problems and motivations for choosing the research topic. The main and specific objectives and research questions were then defined, the adopted methodology was explained, and the publications produced throughout this work were summarized. Finally, the overall structure of the thesis was outlined.

The next chapter presents the background necessary for understanding this work, as well as the related studies connected to the research.

2

Background and Related Works

In this chapter, the concepts necessary for understanding the work developed in this thesis are presented. The technologies used in the RESafety process—namely, iStar4Safety and STPA—require additional explanation. The iStar4Safety language calls for a discussion of Goal-Oriented Requirements Engineering (GORE), as well as iStar 2.0, the language from which iStar4Safety was extended. Safety-critical systems are also described, with one of them explained in detail: the Insulin Infusion Pump System, which is used throughout this document as an illustrative example to demonstrate each step of the RESafety process. STPA, in turn, requires a broader discussion of safety analysis, which is likewise addressed in this chapter.

In addition, the TAM model is presented, serving as the basis for the evaluation questionnaire applied in this research. Finally, related works relevant to this thesis are reviewed.

2.1 Goal-Oriented Requirements Engineering (GORE)

Requirements engineering aims to identify the goals that stakeholders expect the system to fulfill, documenting them in a way that enables analysis, communication, and implementation (Nuseibeh & Easterbrook, 2000). The Goal-Oriented Requirements Engineering (GORE), in turn, can encompass both early and late stages of system development. Stakeholder goals are typically modeled as high-level and abstract objectives during the early requirements phase, facilitating a more suitable elicitation and analysis process for this stage. Unlike the later stages, which are guided by well-defined requirements, early stages in GORE consider a broader range of development options. The models created in the early phases should be progressively refined as the system development evolves (Horkoff & Yu, 2016).

GORE provides a broader perspective than traditional system-centered approaches. It enables the modeling of higher-level concerns within the context of complex systems, considering not only the system itself but also its surrounding environment. GORE focuses on activities that precede the formal specification of software requirements and generally involves goal elicitation, goal refinement, various forms of goal analysis, and the assignment of responsibilities to specific agents (Lapouchnian, 2005). One of its main benefits is its strong support for early-stage

requirements analysis.

Goals define what the system must achieve, describing the rationale for its existence and guiding decisions at multiple organizational levels (Anton, 1996). In the requirements elicitation process, goals evolve from high-level strategic intentions to more concrete operational goals through successive refinements (Nuseibeh & Easterbrook, 2000). Goal reasoning using refinement trees provides traceability from strategic high-level goals to lower-level requirements (Lapouchnian, 2005). Another advantage of goal modeling is its effectiveness in communicating requirements to stakeholders, due to the level of abstraction offered by this approach.

Several goal-oriented requirements modeling languages exist. Among them, iStar 2.0 (Dalpiaz *et al.*, 2016) and iStar4Safety (Ribeiro *et al.*, 2019b) are noteworthy and will be introduced in the following sections.

2.1.1 iStar 2.0

The iStar language, originally proposed by Yu (1995), focuses on the social modeling of actors related to the system and their interdependencies. This abstraction is achieved through the description of relationships among actors. The language has been used for specifying early requirements of various types of systems. It is currently in its version 2.0 (Dalpiaz *et al.*, 2016) (see Table 2.1).

Table 2.1: Comparative table of constructs in iStar 1.0 and iStar 2.0.

Nodes and Links	iStar 1.0	iStar 2.0
Actors	General actors,	General actors
	Roles, positions, and agents	Roles, agents
Actor links	*is-a*	*is-a*
	is-part-of, plays, occupies, covers	*participates-in*
	INS	-
Intentional elements	*Goal, task, resource*	*Goal, task, resource*
	Softgoal	*Quality*
Links between intentional elements	*Means-end, task decomposition*	*Refinement*
	Contribution	*Contribution*
		Qualification, neededBy

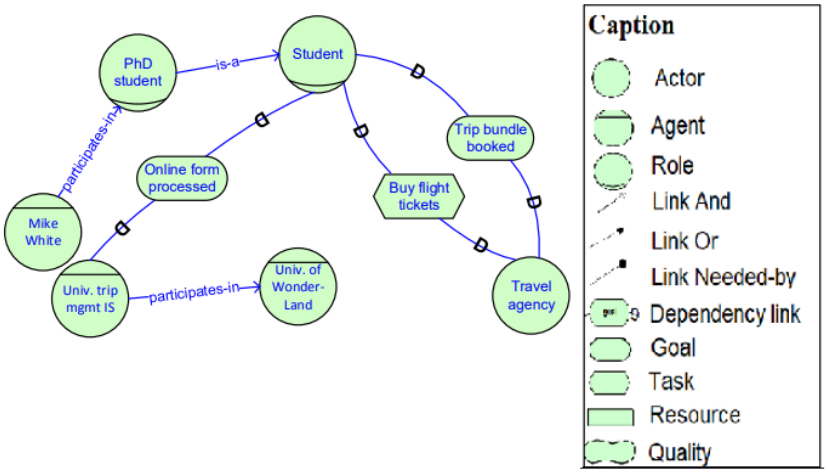
Source: Author (2025).

iStar models (Horkoff & Yu, 2016; Dalpiaz *et al.*, 2016) can be classified into three types, depending on the analyst's intended perspective: Strategic Dependency (SD), Strategic Rationale (SR), or hybrid models.

The Strategic Dependency (SD) model provides a high-level view of the actors in the system and the strategic dependencies that exist among them. An example of an SD

model is presented in Figure 2.1. In this diagram, the scenario represents the process of travel reimbursement. The Student role depends on the Travel Agency to purchase flight tickets and schedule the trip. Additionally, the Student depends on the University to have the online travel form processed.

Figure 2.1: iStar 2.0 SD model example – Travel reimbursement scenario

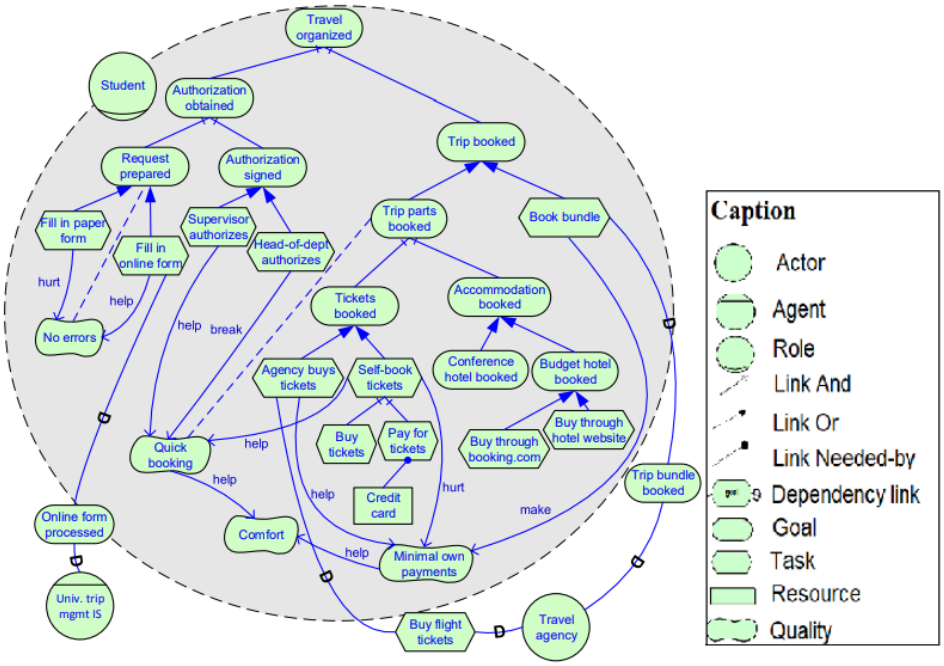


Source: Dalpiaz *et al.* (2016)

.

The Strategic Rationale (SR) model provides a detailed view of the system, allowing the inclusion of all native constructs of the iStar language. It presents the internal reasoning and decision-making logic within the boundaries of each actor. An example of an SR model is shown in Figure 2.2. This model extends the travel reimbursement scenario, illustrating the student's possible options for scheduling the trip, as well as the activities that must be performed by the university in order to process the travel form submitted by the student. It also shows how the travel agency intends to handle the reservation.

Figure 2.2: iStar 2.0 SR model example – Travel reimbursement scenario



Source: Dalpiaz *et al.* (2016)

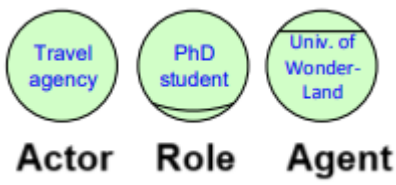
.

The third possible modeling perspective in iStar 2.0 is the hybrid model, which allows a combination of SD and SR elements. In this type of model, some actors may be expanded to show internal reasoning (as in SR models), while others remain abstract and only show their strategic dependencies (as in SD models).

iStar actors, as illustrated in Figure 2.3, can be categorized into three different types:

- **Actor:** A general actor. This type is used when no specific classification is required or intended;
- **Agent:** An actor with concrete, physical manifestations;
- **Role:** An abstract characterization of an actor, representing a position or function in a given context or environment.

Figure 2.3: Types of actors in iStar



Source: Adapted from Dalpiaz *et al.* (2016)

.

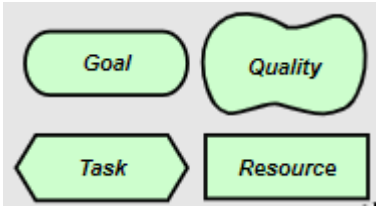
Actors in iStar can also be interconnected through two types of links: ***participates-in*** and ***is-a*** (see Figure 2.1). The *is-a* link represents a generalization/specialization relationship. It is important to note that agents cannot be specialized, as they represent concrete instantiations (Dalpiaz *et al.*, 2016). In contrast, the *participates-in* link denotes relationships that do not reflect generalization or specialization. Any actor may be related to multiple other actors through *participates-in* links.

To emphasize the social and intentional aspects of the language, iStar incorporates intentional elements as core modeling constructs. These elements can appear both inside and outside the actor's boundaries and include:

- **Goal:** Represents a state or outcome that the actor aims to achieve, with clearly defined satisfaction criteria.
- **Quality:** A non-functional attribute that reflects a desirable property or condition the actor wishes to attain. Qualities may express the preferred manner or degree to which other elements should be achieved, thereby influencing goals or tasks.
- **Task:** Represents a concrete action or process that the actor intends to perform or execute.
- **Resource:** A physical or informational entity that the actor requires in order to perform a task or fulfill a goal.

Figure 2.4 illustrates the four types of intentional elements in iStar.

Figure 2.4: Types of intentional elements in iStar



Source: Author (2018).

In an SR model (see Figure 2.2), actors are shown along with their boundaries, which contain intentional constructs that, when combined, represent the strategic rationale of that actor.

In iStar, relationships between actors are modeled through dependencies. A dependency is composed of five elements:

1. ***depender*** – the actor who depends on something;
2. ***dependerElmt*** – the specific element within the depender's model involved in the dependency;

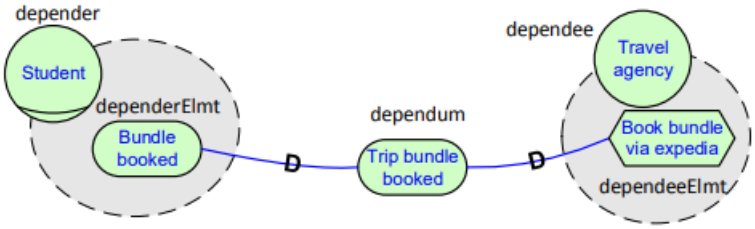
3. ***dependum*** – the object of the dependency (i.e., what is being depended upon: a goal, task, resource, or quality);

4. ***dependee*** – the actor who is expected to fulfill or provide the dependum;

5. ***dependeeElmt*** – the element in the dependee's model that satisfies the dependency.

Figure 2.5 illustrates the structure of a dependency relationship and its components. In this relationship, the depender is the actor who expects the dependee to satisfy a certain condition. This condition is represented by the dependum, which can be any of the four intentional elements previously introduced (goal, quality, task, or resource). The dependerElmt refers to the intentional element within the boundary of the depender actor from which the dependency originates. The dependeeElmt, in turn, is the intentional element within the dependee actor that represents how the actor intends to fulfill the dependency.

Figure 2.5: Elements of a dependency relationship in iStar



Source: Dalpiaz *et al.* (2016).

Intentional elements in iStar can also be interrelated, and the language defines a set of links to support this purpose. Table 2.2 presents the available links in iStar.

Table 2.2: Relationship between iStar links and intentional elements

		Arrowhead pointing to			
		Goal	*Quality*	*Task*	*Resource*
Links originating from	**Goal**	Refinement	Contribution	Refinement	n/a
	Quality	Qualification	Contribution	Qualification	Qualification
	Task	Refinement	Contribution	Refinement	n/a
	Resource	n/a	Contribution	*NeededBy*	n/a

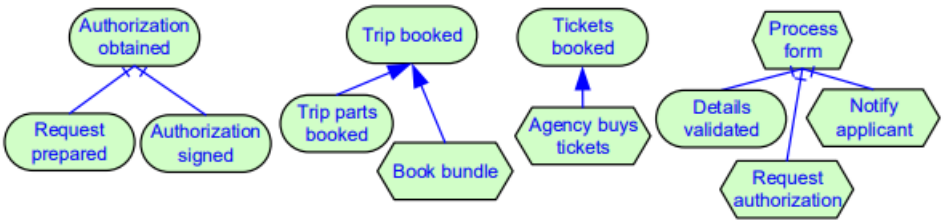
Source: Author (2025).

The refinement links, illustrated in Figure 2.6, connect task and goal elements through a hierarchical structure. In this relationship, a parent element may have up to N child elements and

can participate in only one refinement relationship. The possible refinements are of type AND or OR.

When elements are refined through an AND relationship, all child elements must be satisfied in order for the parent to be fulfilled. In an OR relationship, satisfying at least one child is sufficient to satisfy the parent. If a parent has only one child element, the refinement is considered of type OR by default.

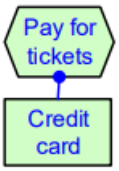
Figure 2.6: Examples of refinement links



Source: Dalpiaz *et al.* (2016).

The NeededBy link, shown in Figure 2.7, connects a task to a resource, indicating the asset required for an actor to perform a specific task (Dalpiaz *et al.*, 2016).

Figure 2.7: Examples of *neededBy* relationships

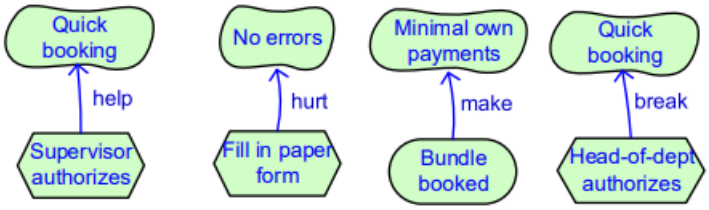


Source: Dalpiaz *et al.* (2016).

The contribution links, illustrated in Figure 2.8, connect intentional elements to qualities, indicating the degree to which the former contribute to the satisfaction of the latter. There are four types of contribution relationships:

- *Help:* Indicates that the element weakly contributes to the satisfaction of the quality.
- *Make:* Indicates that the element sufficiently satisfies the quality.
- *Hurt:* Indicates that the element weakly hinders the satisfaction of the quality.
- *Break:* Indicates that the element strongly prevents the satisfaction of the quality.

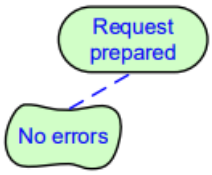
Figure 2.8: Examples of each contribution type in iStar



Source: Author (2025).

When a construct is intended to be performed under a specific quality condition, it is said that the quality qualifies the respective construct. Figure 2.9 illustrates an example in which the goal Requested prepared must be achieved without errors.

Figure 2.9: Example of a qualification relationship



Source: Dalpiaz *et al.* (2016).

2.1.2 iStar4safety

This thesis relies on an extension of iStar for modeling safety requirements, namely iStar4Safety. It was proposed in Ribeiro *et al.* (2019c,b). It is considered suitable for the early modeling of safety requirements in Safety-Critical Systems (Ribeiro, 2019).

- **Safety Goal** – Represented as element A in Figure 2.10. This is a specialization of the *goal* element from iStar 2.0. As a lightweight extension, iStar4Safety adds only a stereotype to the concrete syntax of iStar 2.0 to indicate the element type, in this case *«SafetyGoal»*. A color (e.g., pink) may be used to enhance visual differentiation. Safety goals represent objectives that are critical to the safety of the system being modeled. It is essential that these goals are analyzed and addressed; otherwise, a specific-level accident may occur. Therefore, actors aim to satisfy the safety goals, since their non-fulfillment may result in an accident.

- **Hazard** – Represented as element B in Figure 2.10. This is also a specialization of the *goal* element, functioning as an anti-goal. The stereotype *«Hazard»* is applied to this element. It is recommended to use a distinctive color (e.g., red) to differentiate it from other elements and draw attention within the model. Hazards are obstacles to the

fulfillment of safety goals. When a hazard obstructs a safety goal, it creates a situation of potential accident. Hazards can also be decomposed. A hazard's decomposition represents its root causes and environmental conditions. Environmental conditions are one type of hazard cause. Child hazards are considered causes of the parent hazard.

- **Safety Task** – Represented as element C in Figure 2.10. This is a specialization of the *task* element from iStar 2.0 and, therefore, a specific type of task. It uses the stereotype *«SafetyTask»*. A distinct color (e.g., pink) is suggested for differentiation. The safety task is one of the constructors that can form part of a safety strategy. It is used to model concrete actions taken to mitigate hazards. Leaf hazards should be associated with safety tasks responsible for their mitigation.

- **Safety Resource** – Represented as element D in Figure 2.10. This is a specialization of the *resource* element from iStar 2.0. The stereotype *«SafetyResource»* is applied to identify it as a special type of resource. A color (e.g., pink) may be used for visual distinction. This construct models an asset that is critical to hazard mitigation in the modeled Safety-Critical System (SCS). A safety resource can be associated with a safety task.

- **Obstructs Link** – Represented as element E in Figure 2.10. The *obstructs* link is an n-ary relationship that can relate one parent to one or more children. It indicates that a hazard obstructs the fulfillment of a safety goal. This relationship does not express the degree of obstruction. The link is used from hazards (source) to safety goals (target).

- **Accident Impact Level Property** – This is a property associated with the *Safety Goal* element. The impact level defines how critical a potential accident is in relation to the system's safety.

Figure 2.10: Constructs added by iStar4Safety to the iStar language

Source: Ribeiro (2019).

For further details on the use of the language, as well as examples applied to Safety-Critical Systems (such as an Insulin Infusion Pump and a Railway Crossing Control System), reference can be made to the master's dissertation that presents the development of the language (Ribeiro, 2019), as well as to the scientific papers published on the topic (Ribeiro *et al.*, 2019b,c).

It is also important to highlight that a tool named piStar4Safety[1] was developed to support the modeling of the iStar4Safety extension. It is a customized version of the piStar tool originally designed for modeling with the iStar 2.0 language (Pimentel *et al.*, 2019).

2.2 Safety-Critical Systems

Safety-Critical Systems (SCS) are systems whose failure or unexpected behavior may result in harm or loss of life, property damage, mission failure, and/or environmental harm (Leveson, 1995, 2011). Consequently, investments and engineering effort to ensure their correct functioning are justified and necessary. Following Firesmith (2004), *safety* is the degree to which accidental harm is appropriately addressed—by preventing it, detecting it, responding to it, or adapting to its presence.

Analyzing the safety of SCS involves anticipating accidents and their causes by identifying hazards throughout the system life cycle, seeking to eliminate hazards when feasible or, at minimum, to control or mitigate their consequences. Reliance on historical data and prior experience from other systems is often of limited value because both the systems and the resulting accidents are highly context-specific. Given the criticality of interfaces, it is unlikely that inter-component interactions will manifest identically across different systems (Leveson, 1995). Therefore, safety needs to be integrated across the entire development process and maintained during operation and support (Leveson, 2011).

2.2.1 Foundational Distinctions and Process Implications

It is important to distinguish *safety* from *security*. While safety focuses on freedom from accidental harm, security concerns protection against malicious actions such as unauthorized disclosure, destruction, or modification of information (Berry, 1998). Both qualities can interact, but they require distinct analyses and controls.

Safety analysis must be continuous and kept up to date. Accidents that could have been prevented often occur due to system or environmental changes that violate assumptions captured in the Preliminary Safety Analysis (PSA) and related documentation (Leveson, 2011). Active involvement of stakeholders in safety assessment is recommended, and methods used to derive safety requirements should be learnable, understandable, and applicable by practitioners (Stålhane & Myklebust, 2016).

[1]The tool is available at: `https://www.cin.ufpe.br/~jhcp/pistar/4safety/`.

2.2.2 Stages of Safety Analysis

Safety analysis activities are commonly described across four stages, depending on timing and objectives (Leveson, 1995):

- **Preliminary Safety Analysis (PSA)/Preliminary Hazard Analysis (PHA):** Iterative early-lifecycle identification of critical functions and broad hazards.
- **System Safety Analysis (SSA)[2]:** Refinement of PSA hazards and analysis of how system-level behavior interacts with safety requirements under integration.
- **Subsystem Safety Analysis (SSSA):** Sub-analysis that examines component/subsystem behaviors and their impact on overall system safety.
- **Operating and Support Safety Analysis (OSSA):** Identification of hazards and risk-reduction procedures throughout operation, maintenance, and support phases, lasting until end of life.

2.2.3 Requirements Engineering for SCS

Specification languages support human problem solving by aiding (i) reasoning about particular properties, (ii) construction of software to meet system needs, and (iii) validation that intended qualities are being achieved (Leveson, 2011). Documenting and tracing hazards and their resolutions is a baseline need for any safety program.

Eliciting requirements for SCS is costly due to the need to capture behaviors across interacting subsystems and to accommodate numerous constraints (Broomfield & Chung, 1997). Evidence indicates that many safety-related failures originate from requirements errors rather than coding defects (Leveson, 1995). Requirements specifications are frequently incomplete—omitting unlikely but critical hazards or combinations of events—and may remain ambiguous or inconsistent even when extensive (Firesmith, 2004). Two common software-related safety error sources are (i) inadequate requirements for software interfaces with the wider system and (ii) discrepancies between documented and actually needed requirements (Lutz, 1993).

2.2.4 Integrating Safety and Software Engineering

A *critical* software is any software that can directly or indirectly contribute to a hazardous system state (Leveson, 1995, p. 156). Software is not inherently dangerous, but as a system component it may increase or reduce system safety, including when providing critical data for stakeholder decision-making (Broomfield & Chung, 1997; Lutz, 2000). Effective SCS engineering requires understanding software's role and all its interfaces within the system.

[2]Also termed Safety Hazard Analysis (SHA) in some sources.

Requirements engineering activities play a central role during development and certification—often costly yet essential in SCS contexts (Lutz, 2000). Safety engineering and software engineering perspectives should be addressed jointly within RE to enable active participation and communication across disciplines (Broomfield & Chung, 1997). Achieving a shared terminology remains challenging yet crucial for successful collaboration (Leveson, 1995). Conceptual foundations toward a common vocabulary for RE and safety can be found in Vilela *et al.* (2017a).

2.2.5 Key Safety Concepts

Consistent terminology mitigates communication gaps across heterogeneous teams (Leveson, 1995). The following concepts (adapted from Vilela *et al.*, 2017c,a; Leveson, 1995, 2011; Berry, 1998) are central to early safety analysis:

2.2.5.0.1 Accident. An unplanned and undesired (though not necessarily unexpected) event that results in at least one loss of specified severity. Defining accidents informs prevention and corrective strategies. For an Insulin Infusion Pump (IIP), "patient receives an electric shock" illustrates an accident following a protection failure.

2.2.5.0.2 Hazard. A system state or set of conditions that, together with environmental conditions, will inevitably lead to an accident (Leveson, 1995, p. 177). Example in IIP: *delivery system failure* leading to insulin overdose. Hazards should include only elements necessary and sufficient for an accident. Treatment precedence is: eliminate, reduce, control, and finally minimize damage (Leveson, 1995; Berry, 1998).

2.2.5.0.3 Cause of Hazard. A condition sufficient (alone or in combination) to produce the associated hazard (Berry, 1998). Categories include (Vilela *et al.*, 2018):

- *Procedural*: routine or official practices that can induce hazards (e.g., device sharing leading to infection).
- *Interface*: failures at component interfaces (e.g., faulty sensor reading).
- *Human factor*: human interaction as a major source of safety issues; attribution must avoid blame bias and consider design remedies (Leveson, 2011).
- *Environmental*: external conditions threatening the system (e.g., moisture causing shock).
- *System cause*: system-generated events, including *failures* (software/hardware; mechanical/electronic) with probability levels (frequent, probable, occasional, remote, improbable) and *system misbehavior* at the system level.

2.2.5.0.4 Environmental Conditions. Physical, cultural, demographic, economic, political, regulatory, or technological elements external to the system that can affect its behavior. Some are identifiable early; others emerge as design decisions and analyses evolve.

2.2.5.0.5 Functional Safety Requirements. Requirements that prevent or mitigate the effects of identified failures (e.g., "replace reservoir" to address "broken reservoir" in the IIP).

2.2.5.0.6 Constraints. Engineering decisions imposed as requirements specifying how software must be developed or operate (Vilela *et al.*, 2017c; Firesmith, 2004). High-level safety constraints originate from identified hazards (Leveson, 2011). Constraints introduce development trade-offs and uncertainty (Morandini *et al.*, 2017).

2.2.5.0.7 Obstacles. Behaviors or goals that block the satisfaction of a given goal; duals of goals that define exceptional/undesired conditions and are refined via AND/OR structures (Van Lamsweerde & Letier, 2000; Anton, 1996; Vilela *et al.*, 2017c). In safety contexts, obstacles are often instantiated as hazards.

2.2.5.0.8 Pre/Postconditions. Preconditions restrict adoption of goals to specific contexts; postconditions characterize system state upon goal achievement (Morandini *et al.*, 2017; Anton, 1996).

2.2.5.0.9 Criticality Level of Safety Elements. A predefined scale expressing how strongly an element can contribute to hazards/accidents, considering control over critical functions. Criticality for parent goals should be inherited by their refinements to avoid propagating compromised information.

2.2.5.0.10 Safety Strategies. Mitigations that eliminate or reduce risk, each incurring cost (often resource consumption) (Asnar *et al.*, 2011; noa, 2012). For software failure in IIP, strategies may include alerts and vendor support.

2.2.5.0.11 Safety Resources. Information or physical assets required to realize tasks; in SCS, specialized resources that are critical for safe operation (Dalpiaz *et al.*, 2016; Vilela *et al.*, 2017c).

2.2.5.0.12 Accident Impact Level. Typical categories include *Catastrophic*, *Hazardous/Severe-major*, *Major*, *Minor*, and *No effect* (Leveson, 1995; noa, 2012; Zoughbi *et al.*, 2011). In IIP, insulin overdose is generally *Catastrophic*; underdose may be *Hazardous/Severe-major*.

2.2.6 Documentation, Change, and Traceability

Safety documentation—particularly hazard analysis—must be continuously updated and actively used. Changes in the system or its environment can invalidate assumptions in the PSA and related artifacts, increasing the likelihood of accidents (Leveson, 2011). Sustained stakeholder involvement, clear terminology, and strong traceability between hazards, constraints, and requirements are key enablers for maintaining safety throughout the lifecycle.

2.2.7 Insulin Infusion Pump System

To illustrate the concepts applied throughout this thesis, the Insulin Infusion Pump System—a well-known Safety-Critical System (SCS)—is presented. A malfunction or unexpected behavior in this type of system may result in severe consequences, including financial loss, environmental damage, injury, or death.

As discussed by Zhang *et al.* (2011), the terminology commonly used in safety analyzes often poses challenges to communication and understanding. However, it is essential to apply a systematic and disciplined method to assess how a system might cause harm.

This system was selected due to its level of complexity and criticality, and because it is frequently used in the literature for safety modeling (Martins *et al.*, 2015; Vilela *et al.*, 2017c; Sommerville, 2010). According to the FDA, more than 5,000 adverse events related to the use of insulin infusion pumps were reported in 2008 (Zoughbi *et al.*, 2011). Furthermore, the same example was adopted in previous work (Ribeiro, 2019), in which iStar4Safety—one of the techniques employed in the proposed process—was introduced.

The study adopted in this work is adapted from Martins *et al.* (2015) and supplemented with findings from Zhang *et al.* (2010, 2011), who describe the development of a low-cost infusion pump, a generic model of such systems, and an associated hazard analysis. The level of abstraction in the system model directly influences the depth of the analysis. More detailed models lead to the identification of additional hazards, their causes, and mitigation strategies (Zoughbi *et al.*, 2011).

In this work, the system is used to demonstrate the application of the proposed safety process, without the intention of exhaustively analyzing all functional and safety requirements. Instead, a representative subset is employed, sufficient to validate the process.

The insulin infusion pump is designed to support the treatment of Type 1 Diabetes Mellitus. According to Martins *et al.* (2015), this device automates insulin administration, simulating physiological processes, and reducing user involvement. The pump delivers insulin in two modes: basal (continuous) and bolus (rapid).

A generic insulin pump model, as described by Zhang *et al.* (2010), includes a user interface (LCD display and audible alarms) and several hardware components: a microprocessor, battery, infusion mechanism, catheter and insulin reservoir.

2.2.7.1 General Requirements

The primary function of the insulin infusion pump is to administer rapid-acting insulin doses via a catheter inserted under the skin. The objective is to maintain stable blood glucose levels for patients with Type 1 diabetes.

The pump typically consists of the following:

- A non-rechargeable battery;
- An insulin reservoir (a standard syringe);
- A stepper motor;
- A user interface (LCD screen, four buttons, and an alarm);
- An infusion set;
- Embedded control software.

This system is designed for continuous operation—24 hours a day, 7 days a week—and is suitable for all age groups, provided the user is adequately trained. Cost-effectiveness is also a key requirement; thus, low-cost components (e.g., common syringes) are used.

The pump allows for two insulin delivery types:

- **Basal:** continuous low-dose administration throughout the day.
- **Bolus:** additional, user-triggered doses in response to meals or glucose variations.

The infusion pump controller (representing the embedded software) must fulfill several functions:

- Interpret user commands and input data;
- Maintain and update insulin delivery profiles;
- Recommend bolus doses based on glucose levels;
- Generate and send instructions to the delivery mechanism;
- Display system status for user monitoring;
- Trigger alarms for warnings and errors;
- Store operational data for future review.

External factors such as temperature, sound, pressure, and radiation may interfere with the pump's behavior.

The user is responsible for tasks such as checking configuration settings, replacing the battery, and updating the infusion set. A separate interface should display battery and insulin levels clearly.

The patient can configure and adjust both basal and bolus profiles. The pump manages infusion initiation and control, including stepper motor activation and dose counting.

The controller must monitor sensors, update the display, and ensure safe system reinitialization—critical for avoiding hazards. The device should also provide support features, including manufacturer-assisted training and maintenance.

2.2.7.2 Safety Requirements

Several safety requirements are associated with the use of insulin pumps, stemming from misuse, malfunctions, or environmental conditions. Critical objectives include:

- Ensuring accurate insulin dosing to prevent overdose or underdose. Overdose may result from incorrect configuration, unintended insulin flow, excessive bolus requests, or delivery system failure. Underdose may occur due to incorrect settings, catheter disconnection, reservoir issues, or leakage.
- Preventing electrical shock, which could result from poor insulation, contact with moisture, or component failure.
- Avoiding unexpected reset to factory settings, which may be triggered by software or hardware faults.
- Minimizing infection risks during battery or infusion set replacement, especially if the device is improperly sterilized or shared between users.
- Guaranteeing proper system reinitialization, as failure to do so may lead to life-threatening insulin misadministration.

2.2.7.3 Mitigation Strategies

The following mitigation strategies aim to reduce the likelihood or consequences of hazards:

- Limit configurable dosage ranges on the interface and ensure users are trained—preferably by the manufacturer.
- Require users to inspect the infusion path regularly to detect issues such as leaks or disconnections.

- Issue warnings when bolus is requested without prior food intake.
- Trigger audible alerts for mechanical or delivery system failures.
- Ensure clear communication protocols with the supplier for user support and maintenance.
- Prompt the user to verify settings after system reinitialization.
- Provide documentation on sterilization and sharing policies to prevent contamination.
- Interrupt device use and seek specialized assistance when exposure to moisture or electrical anomalies is detected.

These safety measures are intended to enhance the resilience of the insulin pump system, ensuring that critical failures are either prevented or adequately managed.

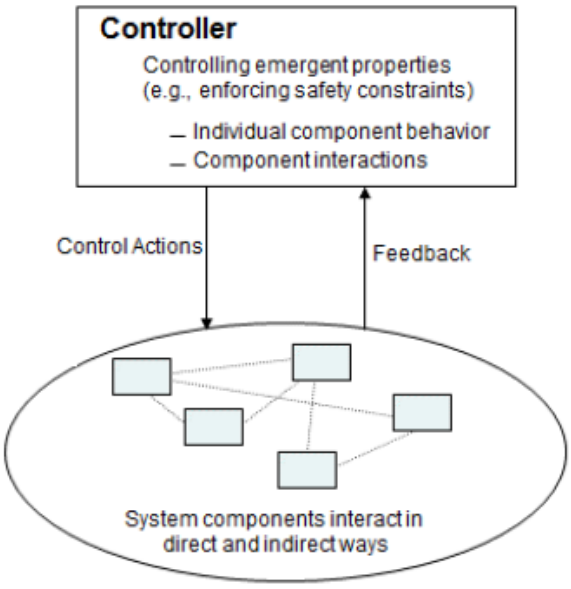
2.3 STPA: System-Theoretic Process Analysis

The *System-Theoretic Process Analysis (STPA)* technique was introduced by Leveson (2011) as part of the *STAMP* model (System-Theoretic Accident Model and Processes). STPA represents a paradigm shift in safety analysis by rejecting the traditional assumption that accidents are primarily caused by component failures. Instead, it focuses on the unsafe interactions among components, which can result in accidents even when individual elements behave as specified.

STPA is grounded in **systems theory**, contrasting sharply with analytical decomposition. While analytical decomposition assumes that understanding individual components in isolation leads to understanding the entire system, systems theory emphasizes that a system's behavior arises from *emergent properties*—behaviors and characteristics that result from interactions among components. As safety is such an emergent property, it cannot be fully understood by analyzing components in isolation. It must be evaluated through both component behavior and inter-component interactions.

This systemic view justifies the introduction of the *controller* in STPA models. In this context, the controller plays a critical role by enforcing constraints on both component behavior and their interactions, ensuring system-wide safety. The controller's role includes evaluating control actions and feedback to prevent unsafe conditions. Figure 2.11 illustrates the standard controller model used in STPA.

Figure 2.11: Standard controller model



Source: Leveson & Thomas (2018).

In STAMP, control loops—comprising control actions and feedback—are fundamental. These loops dynamically adapt to internal and external changes, creating a feedback-driven model of continuous safety assessment. Hierarchical control structures allow upper-level controllers to impose constraints on lower-level processes, aligning with how system-wide constraints are enforced.

Furthermore, Leveson emphasizes the importance of **defining system boundaries**. These boundaries are not inherent but are design choices. Once defined, they determine what is considered part of the system and what is treated as part of the external environment. The recommendation is to define the boundary around components that designers can reasonably control (Leveson, 2011).

Although studies have shown that STPA can identify all hazard scenarios found by techniques such as HAZOP (McDermid *et al.*, 1995) and FTA (Ruijters & Stoelinga, 2015), and even additional ones, its adoption remains lower compared to traditional hazard analysis techniques (Leveson & Thomas, 2018). Nonetheless, STPA is applicable across various development phases, from architecture to detailed design and implementation.

The concept of a controller in STPA is broad. Unsafe interactions and component failures can be mitigated through appropriate design. Furthermore, safety may be enforced via procedural and organizational controls—such as culture, policies, and human behavior.

STPA consists of four main steps, each contributing a distinct layer to the safety analysis process:

- **Define the purpose of the analysis:** Identify unacceptable losses, hazards that can lead to those losses, and the high-level constraints necessary to prevent them.

- **Model the control structure:** Represent the system's control relationships and feedback mechanisms that influence behavior and safety.
- **Identify Unsafe Control Actions (UCAs):** Analyze control actions to determine when they might lead to hazards, considering different failure modes and timing issues.
- **Identify Loss Scenarios:** Examine how unsafe control actions and failures in the control structure might occur under specific circumstances, identifying causal scenarios.

Each of these steps is detailed in the following subsections.

2.3.1 Defining the Purpose of the Analysis

The first step involves identifying the foundational elements of the analysis. This includes:

1. Identifying unacceptable losses
2. Identifying system-level hazards
3. Deriving system-level safety constraints
4. Optionally refining hazards

A **loss** is defined as any unacceptable outcome to humans or stakeholders and is often associated with accidents or adverse events. Losses can be prioritized based on stakeholder input, starting with the identification of stakeholders and their values, objectives, or intended system uses. These elements are then translated into concrete losses (e.g., loss of life, environmental harm, mission failure). An example for an IIPS could be **L-1 = Hypoglycemia**.

Next, **hazards** are identified. These are system states or conditions that, combined with specific environmental contexts, may lead to a loss. To define them properly, it is necessary to first establish the system boundary—i.e., the set of components under the designers' control. Hazards are described using the format:

<System> <Unsafe condition> <Link to loss>

For example:

H-1 = The insulin pump allows free flow of insulin [L-1]

Hazards should remain high-level in the early stages of analysis to avoid an early focus on component-level behavior. Each hazard may be linked to one or more losses. In this example, [L-1] corresponds to the loss Hypoglycemia, as previously defined.

System-level safety constraints are then derived to mitigate the hazards. Typically, constraints are the inverse of a hazard condition. For instance, the constraint corresponding to H-1 would be:

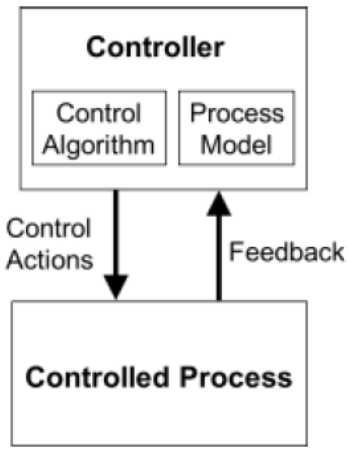
SC-1 = The insulin pump must block the free flow of insulin [H-1]

Each constraint should address system behavior holistically and avoid detailing specific implementation-level solutions.

2.3.2 Modeling the Control Structure

As illustrated in Figure 2.12, control systems typically consist of various loops, such as control loops and feedback loops. A controller, in general, performs control actions over a controlled process.

Figure 2.12: Generic control loop

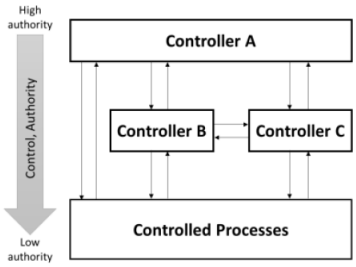


Source: Leveson & Thomas (2018).

The controller contains a control algorithm and a process model. The control algorithm represents the controller's decision-making logic, while the process model captures the internal beliefs (about the controlled process, the system, or the environment) used in making those decisions. Failures in any part of this control loop can result in Unsafe Control Actions (UCAs).

The controlled process can also provide feedback to the controller. When multiple control loops interact, they form a hierarchical control structure, which is typical in most systems. An example of such a structure is shown in Figure 2.13.

Figure 2.13: Generic hierarchical control structure



Source: Leveson & Thomas (2018).

In the vertical dimension of a control structure, the hierarchy represents that higher-level controllers impose constraints on the behavior of lower-level entities. The entity directly beneath another is likely the controlled process of the one above.

Modeling the control structure is an iterative process that typically starts at a high level, especially when knowledge of the system is still limited—as is common in early development stages. Even when modeling an existing system or one with well-defined requirements, starting with a high-level view supports a holistic understanding that can foster innovative design alternatives.

It is recommended to initially model subsystems essential for satisfying system-level constraints and preventing previously identified hazards. A subsequent level of refinement involves identifying the controlled processes within the structure.

As the control structures evolve, responsibilities can be assigned to each controller or system entity. According to Leveson & Thomas (2018), these responsibilities should serve as "a refinement of the system-level constraints."

An example responsibility for an insulin pump controller might be:

R-1: Block high doses of insulin [SC-1]

Control actions for each controller can be derived from these responsibilities. For instance, for R-1, the control action might be "Regulate insulin dosage," which the insulin pump controller would issue to the pump mechanism. Feedbacks can then be defined based on these control actions and responsibilities. It is important to determine what information the controller needs to make accurate decisions. Such information is received via feedback from the controlled process and should be precise to ensure the reliability of the controller's process model. For R-1, a relevant feedback might be the volume of insulin administered, which should be reflected in the process model.

As responsibilities are further refined, so too are the control structures, becoming increasingly detailed. However, physical implementation details and structural completeness are best addressed in later stages of the analysis, such as during the scenario definition phase.

2.3.3 Identifying Unsafe Control Actions (UCAs)

After modeling the control structure, the next step is to evaluate the control actions to determine which of them may be unsafe. An unsafe control action (UCA) is one that, under certain conditions and worst-case environmental scenarios, can lead to a hazard (Leveson & Thomas, 2018).

To identify UCAs, each control action should be examined in terms of the following four conditions:

1. Not providing the control action causes a hazard.
2. Providing the control action causes a hazard.
3. Providing the control action too early, too late, or in the wrong order causes a hazard.
4. The control action is applied for too long or stopped too soon (this is relevant for continuous rather than discrete control actions).

A table like the hypothetical example shown in Table 2.3 can be used to organize the analysis. Each row lists a control action, and the columns represent the four unsafe scenarios. If a scenario applies, it should be documented in the corresponding cell along with the related UCA ID and the associated hazards.

Table 2.3: Example UCAs for an Insulin Pump Controller System

Control Action	**Not providing causes hazard**	**Providing causes hazard**	**Too early, too late, out of order**	**Stopped too soon, applied too long**
Regulate insulin dosage	UCA-1: The insulin pump controller (IPC) fails to regulate insulin dosage during infusion [H-1]	—	UCA-2: The IPC delays insulin dosage control, leading to constant flow [H-1]	—

Source: Author (2025).

The literature emphasizes maintaining traceability by linking each UCA to the associated hazards. Each UCA should also include the context in which the control action becomes unsafe. If a control action is always unsafe, it should not be included in the system design.

To express context, terms like “when,” “while,” or “during” are recommended. A well-defined UCA should contain the following five elements:

1. **Source:** the controller issuing the action (e.g., Insulin Pump Controller)
2. **Action type:** one of the four categories above (e.g., fails to provide)
3. **Control action:** as defined in the control structure (e.g., regulate insulin dosage)

4. **Context:** the operational condition (e.g., during infusion)

5. **Hazard link:** reference to the related hazard(s) (e.g., [H-1.1])

The order of these elements is flexible. Including clear contextual information in the UCA descriptions facilitates the derivation of requirements and the identification of relevant scenarios.

2.3.3.1 Identifying Controller Constraints

Controller constraints specify the behaviors that controllers must follow to prevent Unsafe Control Actions (UCAs) (Leveson & Thomas, 2018). These constraints are typically formulated as the inverse of the corresponding UCA. Table 2.4 illustrates common examples of how constraints are defined.

Table 2.4: Examples of Controller Constraints for the Insulin Pump Control System

Unsafe Control Actions (UCAs)	**Controller Constraints**
UCA-1: The insulin pump controller (IPC) fails to regulate insulin dosage during infusion [H-1.1]	The IPC must regulate the insulin dosage during infusion. [UCA-1]
UCA-2: The IPC provides insulin dosage control too late during infusion, leading to a continuous flow [H-1.1, H-2.31]	The IPC must not delay insulin dosage control when the insulin flow is continuous. [UCA-2]

Source: Author (2025).

As shown in Table 2.4, each identified UCA should be analyzed, and a corresponding controller constraint should be defined. These constraints are essential for guiding the design and implementation of safe controller behavior. Each constraint must explicitly link to the UCA it addresses to ensure traceability and support for hazard mitigation.

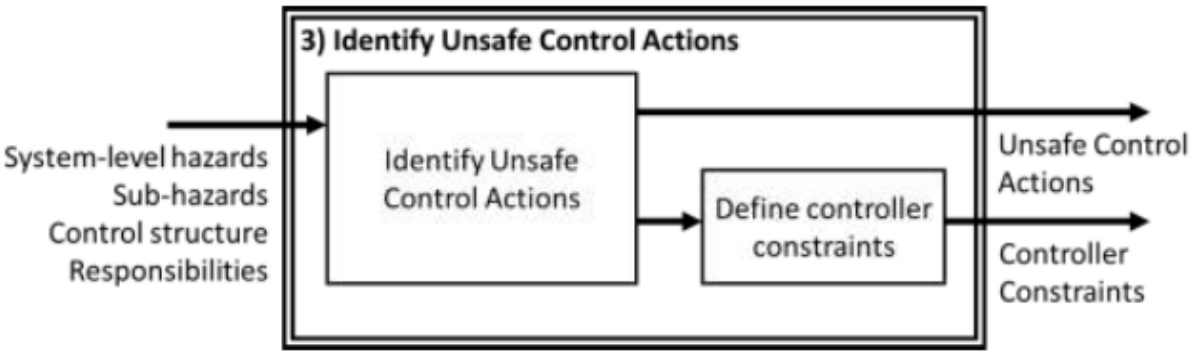
2.3.3.2 Analysis of Control Actions: Inputs and Outputs

The analysis of control actions involves identifying key inputs and expected outputs. The primary inputs include: (1) *System-level hazards*, (2) *Sub-hazards*, (3) *Control Structure*, and (4) *Responsibilities*. These elements form the foundation for evaluating the safety of control actions within the system.

The outcomes of this analysis are twofold: (1) *Unsafe Control Actions (UCAs)*, and (2) *Controller Constraints*. These outputs are critical for guiding the refinement of system behavior to ensure safety.

Figure 2.14 provides a high-level overview of this phase, depicting how inputs are transformed into actionable safety elements through the control action analysis process.

Figure 2.14: Overview of inputs and outputs in the control action identification phase



Source: Leveson & Thomas (2018).

2.3.4 Identifying Loss Scenarios

The fourth and final step of STPA focuses on identifying *loss scenarios*, which are causal factors that can lead to hazards and unsafe control actions (UCAs).

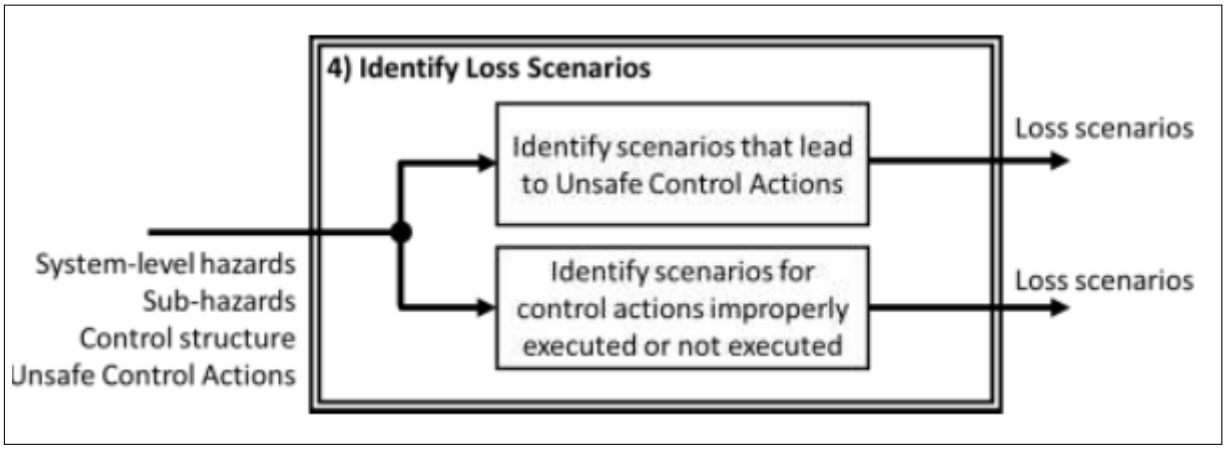
According to Leveson & Thomas (2018), two main types of loss scenarios must be considered:

1. Why would unsafe control actions occur?
2. Why would control actions be executed improperly or not executed at all, thereby leading to hazards?

To address the first question, analysts must identify scenarios in which control actions—though designed to be safe—become unsafe due to specific conditions. For the second, scenarios should be investigated in which control actions are issued incorrectly or fail to be issued, contributing to hazard realization. More detailed guidance can be found in Leveson & Thomas (2018).

Figure 2.15 summarizes the key inputs and outputs of this loss scenario identification phase. Note that this phase builds upon previously defined elements, including *System-level hazards*, *Sub-hazards*, the *Control Structure*, and identified *UCAs*. The primary output of this step is the set of *Loss Scenarios*.

Figure 2.15: Overview of inputs and outputs for the loss scenario identification phase



Source: Leveson & Thomas (2018).

2.4 Technology Acceptance Model (TAM)

The Technology Acceptance Model (TAM), introduced by Fred Davis in 1986 as part of his doctoral research (Davis, 1989; Lai, 2017), was developed to investigate the reasons behind users' acceptance or rejection of information technologies. Based on the Theory of Reasoned Action (TRA), TAM seeks to answer two fundamental questions: (1) What factors influence the acceptance of a technology? and (2) How do system characteristics affect user decisions?

At the heart of TAM are two key constructs: **Perceived Usefulness (PU)** and **Perceived Ease of Use (PEOU)**. PU refers to the extent to which a person believes that using a system will enhance their job performance, while PEOU reflects the belief that using the system requires minimal effort. These factors jointly shape the user's **Behavioral Intention (BI)** to use the system, which in turn predicts actual usage.

TAM has become one of the most influential and widely used models in IT acceptance research, especially in software engineering contexts. As reported by Börstler *et al.* (2024), TAM and its extensions were present in 61.7% of the reviewed studies—far surpassing Innovation Theories, which appeared in 21.3% of cases. Moreover, TAM has been applied in the context of at least seven different software engineering models, demonstrating its adaptability and relevance across varied scenarios.

Several previous studies, including Opdahl & Sindre (2009); Peraldi-Frati *et al.* (2019); da Silva & Oliveira (2020); Soares & do Nascimento (2014), have successfully applied TAM or its variants to assess user perceptions regarding new software tools, particularly with respect to usefulness and usability.

In this thesis, TAM is adopted as the foundation for evaluating user perceptions of the RESafety process. The core constructs guiding the design of the evaluation instruments are

described below:

- **Perceived Usefulness (PU):** The extent to which a person believes that using a particular system will enhance their job performance.
- **Perceived Ease of Use (PEOU):** The degree to which a person believes that using the system will require little or no effort.
- **Behavioral Intention (BI):** The extent to which a person intends to use the system in the future, influenced by both PU and PEOU.

These constructs are central to TAM and will serve as the basis for the evaluation conducted in this study.

The following section of this chapter presents the related works that support the development of this thesis.

2.5 Related Works

2.5.1 Sharifi et al.

Sharifi et al. (2022) propose a requirements-based guideline to improve the certification process of FinTech systems, particularly those involving digital asset custody. The authors argue that financial technologies increasingly operate as socio-technical systems with significant safety, privacy, and compliance risks, making traditional audit-based certification insufficient. To address this, they combine the User Requirements Notation (URN)—which integrates Goal-oriented Requirement Language (GRL) and Use Case Maps (UCMs)—with the System-Theoretic Process Analysis (STPA) technique to guide the early modeling and certification of FinTech systems. Their guidelines are organized into five steps: (1) identifying and classifying stakeholders into governance, regulatory, business, and technical domains, (2) creating a strategic dependency model using GRL to elicit goals and dependencies, (3) refining functional goals through UCMs to build process-level traceability, (4) performing STPA, subdivided into defining system boundaries, control structures, UCAs, loss scenarios, and safety constraints, and (5) constructing assurance cases based on STPA-related argumentation. The resulting framework enables a clear linkage from system-level goals to safety constraints and certification artifacts, ensuring traceability from regulatory concerns to technical requirements.

Unlike RESafety, which integrates iStar4Safety (an extension of iStar 2.0) and STPA within a BPMN-modeled iterative process aimed at generalizing early safety analysis across diverse safety-critical systems, the FinTech guideline by Sharifi et al. focuses on a domain-specific application. Their proposal primarily targets the certification and assurance perspective, emphasizing compliance communication rather than the systematic elicitation and refinement of safety requirements. Moreover, while RESafety formalizes traceability between goals, UCAs,

hazards, and safety requirements, the FinTech guideline extends URN to include the generation of assurance cases, which are not covered in RESafety. Thus, both approaches align in integrating goal-oriented modeling and STPA to support early safety reasoning, yet differ in their scope and purpose: Sharifi et al. emphasize regulatory certification for FinTech systems, whereas RESafety provides a process-oriented and generalizable method for eliciting and refining safety requirements in any safety-critical domain.

2.5.2 Vilela et al.

SARSSi* (Safety Requirements Specification Method based on STAMP/STPA and iStar; Vilela *et al.* (2019)) proposes a model-based method that combines the System-Theoretic Process Analysis (STPA) technique with the iStar modeling language to support the early specification of safety requirements for safety-critical systems. The main goal of SARSSi* is to represent the results of the STPA safety analysis directly in iStar, facilitating the visualization and traceability of safety-related information within intentional models. Importantly, SARSSi* does not extend the iStar language nor introduce new constructs or relationships; it employs the original iStar 1.0 notation. In contrast, RESafety is based on iStar4Safety, an extension of iStar 2.0, which includes explicit constructs for modeling hazards, safety goals, mitigation strategies, and obstruct links, providing a richer and semantically grounded representation of safety concepts.

SARSSi* defines a six-step iterative process. In the first step, accidents are identified and represented as goals in an iStar Strategic Dependency (SD) model. In step two, a Strategic Rationale (SR) model is developed, in which hazards, their causes, and environmental conditions are modeled as goals, expanding the intentional structure initiated in the SD. In step three, the hierarchical control structure is defined, using the actors already modeled in the SD and SR as components, and encouraging decomposition into software modules during the first iteration. This step also suggests updating the SD and SR models, whereas RESafety postpones such updates until the final iteration stage (step 6). The fourth step identifies control flaws, which conceptually correspond to RESafety's step 4, where Unsafe Control Actions (UCAs) are derived. In step five, SARSSi* models safety requirements as iStar tasks that mitigate hazards—similar to RESafety's notion of mitigation strategies—although without using explicit links to STPA artifacts. Finally, step six includes a consistency verification guideline, ensuring the completeness and coherence of the models, comparable to the consistency check proposed in RESafety.

SARSSi* shares several methodological characteristics with RESafety, such as its iterative nature, the use of identifiers for traceability, and the mapping between actors in iStar and components in the STPA control structure. However, there are key differences. SARSSi* treats environmental conditions as distinct modeling elements, while RESafety implicitly considers them through causal reasoning in loss scenarios. More importantly, SARSSi* focuses primarily on the representation of STPA results using iStar, whereas RESafety provides an integrated and process-oriented framework that explicitly aligns the reasoning steps of STPA

with the modeling capabilities of iStar4Safety. RESafety also formalizes the integration through six BPMN-modeled steps, includes rules for model refinement and consistency maintenance, and supports the traceability of UCAs, hazards, loss scenarios, and safety requirements across artifacts.

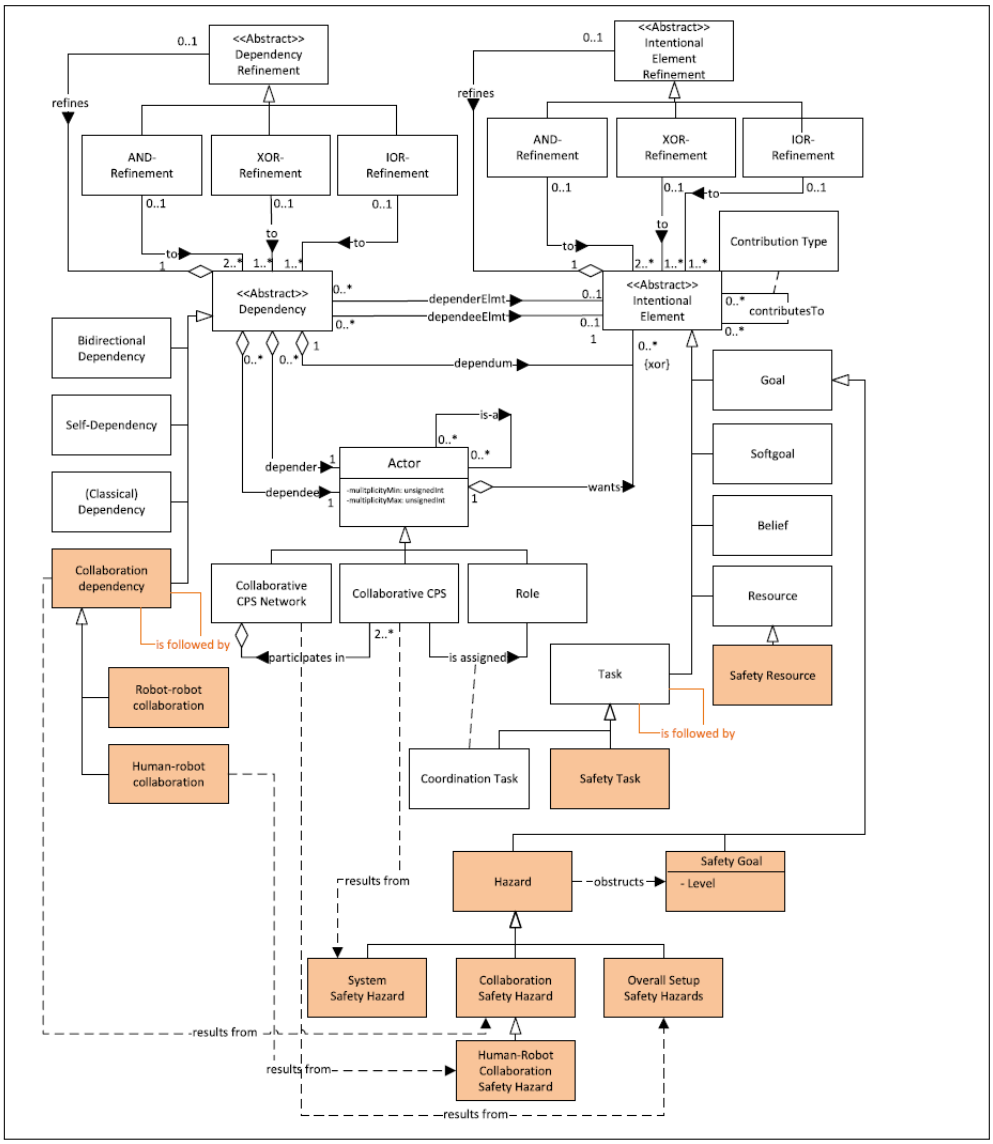
Therefore, while both SARSSi* and RESafety seek to bridge Requirements Engineering and Safety Engineering through the integration of goal-oriented modeling and system-theoretic hazard analysis, RESafety extends this integration by offering a semantically enriched, structured, and generalizable process that leverages the expressiveness of iStar 2.0-based iStar4Safety to support safety requirements modeling across any safety-critical domain.

2.5.3 Manjunath et al.

Early Model-Based Safety Analysis for Collaborative Robotic Systems (Manjunath *et al.*, 2025) proposes a model-based methodology for early safety assessment in human–robot collaboration (HRC). The authors argue that as robots increasingly cooperate with humans in industrial environments, ensuring safety during the early stages of system design becomes critical. Their approach extends the Goal-oriented Requirement Language (GRL) to model safety hazards and mitigation strategies within collaborative robotic settings.

The proposed metamodel—illustrated in Figure 2.16—integrates and expands prior works, including iStar4Safety (Ribeiro *et al.*, 2019b), by introducing new constructs such as collaboration dependencies (representing human–robot and robot–human collaborations), hazard types (system safety hazards, collaboration safety hazards, and overall setup safety hazards), safety tasks, safety goals, and safety resources, in addition to the obstructs link between hazards and safety goals. It is important to highlight that the authors categorize hazards according to aspects intrinsic to human–robot collaboration, which differs substantially from RESafety, designed to be a generalizable process applicable to any safety-critical system.

Figure 2.16: Extended GRL-based metamodel supporting model-based safety analysis for human–robot collaboration.



Source: Manjunath *et al.* (2025).

The methodology was evaluated through an industrial case study and a controlled experiment. Results demonstrated that the approach effectively supports the identification and documentation of safety goals and tasks, helping analysts reason about safety requirements in complex collaborative environments. The experiment confirmed its usefulness and ease of use, showing statistically significant improvements in participants' accuracy and confidence.

Overall, Manjunath *et al.* (2025) advance the integration of goal modeling and safety analysis by offering a systematic and empirically validated framework for early safety assessment of collaborative cyber-physical systems. Their approach aligns closely with the rationale of the RESafety process, particularly in emphasizing early modeling of safety concerns and the integration of goal-oriented representations with safety reasoning. However, while their work focuses on extending GRL to incorporate safety constructs within the specific context of

human–robot collaboration, RESafety proposes a broader and more methodologically structured integration between Requirements Engineering (RE) and Safety Engineering (SE). Specifically, RESafety combines iStar4Safety—a safety-oriented goal modeling language—with the STPA technique to establish explicit traceability between system goals, unsafe control actions, hazards, loss scenarios, and derived safety requirements. Moreover, RESafety is organized into seven iterative BPMN-modeled steps, encompassing not only modeling but also systematic hazard identification, safety constraint definition, and model updating activities, thereby providing stronger process guidance. In contrast, Manjunath *et al.* (2025)'s proposal focuses primarily on the representational aspects of safety goals and collaboration dependencies, without defining a detailed stepwise procedure for integrating hazard analysis results into goal models. Thus, while both approaches contribute to bridging the gap between goal modeling and safety analysis, RESafety extends this integration by establishing a process-oriented framework that enhances traceability, communication between RE and SE domains, and repeatability of early safety analysis, while also enabling the modeling of safety requirements for any type of safety-critical system.

2.5.4 Comparative Table

Table 2.5 summarizes the comparative analysis of the selected approaches across several analytical dimensions. Each group of criteria represents a distinct aspect of methodological design and maturity. The General category identifies the domain of application, main objectives, and modeling languages used. Although the analyzed works apply to distinct contexts—FinTech, generic safety-critical systems, and collaborative robotics—all rely on goal-oriented or intentional modeling notations (iStar, GRL, or their extensions) as the foundation for linking functional and safety concerns.

The Modeling and Safety Integration category shows how each approach represents safety-related elements—such as constraints, unsafe control actions, loss scenarios, and responsibilities—and whether it integrates or extends a safety analysis technique such as STPA. Three of the analyzed works (Sharifi *et al.* (2022), Vilela *et al.* (2019), and RESafety) employ STPA as their analytical basis but differ in scope and integration level. Sharifi *et al.* (2022) combine STPA with URN-based models (GRL and UCM) to support structured safety reasoning for FinTech certification. SARSSi (Vilela *et al.*, 2019) integrates STPA outcomes directly into iStar 1.0 goal models, enabling reasoning about hazards and control flaws. In contrast, Manjunath *et al.* (2025) do not use STPA; instead, they extend GRL to represent collaboration-specific safety hazards, safety goals, and mitigation strategies in human–robot interaction contexts. Finally, RESafety formalizes the integration of iStar4Safety (based on iStar 2.0) with STPA through a structured and iterative process, ensuring explicit traceability among system goals, unsafe control actions, hazards, loss scenarios, and safety requirements.

The Process and Structure group describes the organizational aspects of each method—defined

steps, iteration, traceability, and modeling guidance. Sharifi *et al.* (2022), Vilela *et al.* (2019), and RESafety define explicit procedural steps, whereas Manjunath *et al.* (2025) focus mainly on representational extensions rather than prescribing a structured sequence of activities. Sharifi *et al.* (2022) define five guideline-based steps, SARSSi establishes a six-step process integrating STPA reasoning into iStar models, and RESafety structures six iterative BPMN-modeled steps combining iStar4Safety and STPA. Among these, only SARSSi and RESafety explicitly support iteration and feedback between modeling and safety analysis. Traceability is partially supported by most approaches, but RESafety uniquely enforces bidirectional and systematic traceability among all safety-related artifacts.

The Evaluation and Validation category indicates how each approach was assessed. Sharifi *et al.* (2022) validated their guideline through a FinTech case study; Vilela *et al.* (2019) validated their approach through the modeling of a real IIPS development case, but did not conduct empirical evaluations; Manjunath *et al.* (2025) performed both an industrial case study and a controlled experiment to examine the understandability and usefulness of their extended GRL models. It is important to highlight that RESafety was validated through the implementation of a real medication-delivery system operated by a robotic arm, and evaluated through a TAM-based survey and expert interviews to analyze perceived usefulness and ease of use in a safety-critical context.

Finally, the Tool and Support group reflects the degree of automation and documentation provided. All methods provide at least partial support for modeling or verification activities. Sharifi *et al.* (2022) and RESafety are distinguished by the generation of structured artifacts—assurance cases in the former and systematically derived UCAs, hazards, loss scenarios, and safety requirements in the latter. Vilela *et al.* (2019) include a consistency-checking guide for model verification, while Manjunath *et al.* (2025) provide representational extensions to GRL but do not define consistency mechanisms or automated reporting.

Table 2.5: Comparative analysis of related works, using Y (Yes), N (No), and P (Partial).

#	Group	Criterion	Sharifi (2022)	SARSSi* (2019)	Manjunath (2025)	RESafety (2025)
1	General	System target / domain	FinTech (digital asset custody)	Safety-critical systems (general)	Collaborative robotic systems (HRC)	Safety-critical systems (general)
2		Main objective / focus	Improve certification via requirements-based guidelines	Integrate STPA results into iStar models	Extend GRL for early safety modeling in HRC	Integrate iStar4Safety with STPA in an iterative process for safety analysis
3		Modeling language used	URN (GRL + UCM)	iStar 1.0	GRL (extended)	iStar4Safety (based on iStar 2.0)

Continued on next page

Table 2.5 (continued)

#	Group	Criterion	Sharifi (2022)	SARSSi* (2019)	Manjunath (2025)	RESafety (2025)
4	Modeling and Safety Integration	Language extension	P (uses URN views, no new syntax)	N (uses iStar 1.0 as-is)	Y (extends GRL with collab. and safety constructs)	P (reuses iStar4Safety; no new constructs)
5		Integration with safety analysis technique	Y (STPA)	Y (STPA)	N	Y (STPA)
6		Modeling of safety constraints	Y	Y	Y	Y
7		Modeling of unsafe control actions (UCAs)	Y	P (control flaws)	N	Y
8		Modeling of loss scenarios	Y	N	N	Y
9		Modeling of safety requirements	Y	Y	Y	Y
10		Modeling of responsibilities	P (stakeholder domains)	P (actors and control structure)	P (human/robot roles)	P (controllers, processes)
11	Process and Structure	Process steps	5 steps	6 itreative steps	Not stepwise; focuses on metamodel	6 iterative steps
12		Iterative process	N	Y	N	Y
13		Traceability between safety elements	P	Y (IDs link elements)	P	Y (explicit between goals, hazards, UCAs, LSs, SRs)
14		Guidelines provided	Y (detailed, certification-oriented)	Y (step-by-step)	Y (conceptual)	Y (structured BPMN process)
15		Structured process documentation	Y (guideline + assurance)	Y (six-step + check)	P (method description only)	Y (BPMN process and artifacts)
16	Evaluation and Validation	Type of evaluation	Case study (FinTech)	Not evaluated	Industrial case + controlled experiment	TAM-based survey + expert interviews
17		Evaluation domain	FinTech (digital custody)	Not evaluated	Industrial HRC	Medical (insulin pump) and robotic (med. delivery)
18	Tool and Support	Tool support	P (URN tools, e.g., jUCMNav)	Y (iStar tools: Pistar/OpenOME)	P (prototype; no dedicated tool)	P (prototype in development)
19		Consistency checking	N (no check)	Y (manual check guide)	N	P (manual; automation planned)
20		Documentation and reporting	Y (assurance cases)	P (structured models, no automation)	P (diagrams, no reporting)	Y (each step outputs artifacts for next steps)

Source: The author (2025).

2.5.4.1 Discussion

The comparative analysis summarized in Table 2.5 shows that, among the examined approaches, SARSSi* is the one most closely aligned with RESafety in scope and intent. Both

approaches seek to integrate goal-oriented modeling with safety analysis, and both rely, directly or indirectly, on concepts derived from STPA to support early identification of safety-related concerns. However, despite this conceptual convergence, important structural and methodological differences distinguish the two approaches.

Similar to RESafety, SARSSi* incorporates STPA-derived concepts into iStar models. However, SARSSi* uses iStar 1.0 without extensions, while RESafety adopts iStar4Safety, based on iStar 2.0, which provides dedicated constructs for early safety analysis, such as safety goals, hazards, safety tasks, and safety resources, aligned with STPA artifacts. This notational difference improves the clarity, expressiveness, and traceability of safety concepts, particularly in the resulting models.

Continuing the comparison, it becomes evident that RESafety provides a more clearly explained and detailed process. Its six iterative steps are grounded in STPA's system-theoretic perspective and produce a set of artifacts that support the analysis, such as losses and hazard definitions, Unsafe Control Actions (UCAs), Loss Scenarios (LSs), Controller Constraints, and textual safety requirements. Together, these artifacts enable traceability among the safety-related elements, whereas SARSSi* supports this traceability only partially.

A comparison between the two approaches also reveals important differences in how safety constraints are modeled. In SARSSi, safety constraints emerge implicitly through mitigation strategies that aim to reduce or eliminate hazards. In RESafety, by contrast, these elements are explicitly defined, together with the responsibilities associated with enforcing them, which are assigned to the relevant actors. Responsibilities are therefore only partially represented in SARSSi, mainly through the logical behavior of actors and the elements of the control structure, rather than as first-class modeling constructs.

Regarding Unsafe Control Actions (UCAs), RESafety models them explicitly, following STPA. In SARSSi, the corresponding concepts appear as control flaws, which are represented as additional hazards to be incorporated into the analysis during Step 2. This distinction affects the chain of reasoning used to derive safety requirements.

Another important difference concerns the representation of Loss Scenarios, a key STPA artifact that characterizes the specific conditions under which UCAs may occur. SARSSi neither cites nor models Loss Scenarios. The absence of this element may lead to a less refined analysis, since Loss Scenarios help identify the precise points in the control structure where unsafe behavior may arise. RESafety, in contrast, models Loss Scenarios explicitly and uses them as the basis for deriving safety requirements.

Safety requirements are also treated differently. In RESafety, safety requirements are explicitly modeled and systematically linked to the Loss Scenarios they are intended to mitigate. In SARSSi, safety requirements are modeled as tasks that may be associated with environmental conditions or hazards, depending on the refinement strategy proposed in the method.

Although both SARSSi and RESafety adopt six procedural steps, the nature of these steps differs significantly. SARSSi provides guidelines indicating how to construct iStar 1.0 SD

and SR models, while RESafety includes detailed BPMN models that describe how to build the iStar4Safety models as well as the remaining STPA-based artifacts. Both approaches are iterative and and both provide guidance for structuring the analysis.

The two methods also differ in terms of evaluation. SARSSi does not present a formal evaluation, whereas RESafety includes an empirical assessment with experts in safety and requirements engineering. Regarding tool support, SARSSi can be implemented using any tool that supports iStar 1.0 modeling. For RESafety, dedicated tooling is still under development, although Section 3.3 discusses existing tools that can support the approach and the management of its artifacts.

Finally, consistency checking is explicitly addressed in Step 6 of SARSSi, which specifies what is necessary to verify the coherence of the models. Although RESafety does not provide a standalone consistency-checking guideline, it partially satisfies this requirement through the clarity of its steps, which specify what must be produced and reviewed at each stage. As for documentation and reporting, SARSSi was classified as providing partial support, as it produces only iStar SD and SR models, consistent with its objectives. RESafety, on the other hand, generates a broader set of artifacts that together form a comprehensive safety analysis document intended to be maintained and updated throughout the system's lifecycle.

2.6 Chapter Summary

This chapter presented the concepts and technologies necessary for a better understanding of this thesis. In addition, related works relevant to this research were discussed. The next chapter introduces the RESafety process, its BPMN modeling, and the illustration of each step through a safety-critical system.

3

ReSafety: the process

In this chapter, the RESafety process for modeling safety requirements and generating the safety analysis artifact is described. The chapter begins with an overview and the overall BPMN model, outlining each step in a procedural manner. A detailed explanation of the six steps then follows, each accompanied by a step-by-step representation using the corresponding BPMN model, as well as an illustrative example applied to the IIP System. Finally, two proposals for alternative uses of the RESafety approach are presented.

3.1 Process Description

Given the need for analysts to elicit and model safety requirements, this work proposes a process designed to support the early identification and specification of such requirements.

It is important to note that, if necessary, the process also supports the modeling of other early system requirements. This is enabled by the iStar4Safety language, which is conservative in nature—that is, it preserves all constructs of the original iStar 2.0 language while incorporating additional elements specific to safety analysis.

The iterative nature of the process facilitates the re-evaluation of the system whenever new safety requirements are identified. In other words, adding new features or components to the current analysis will alter its context—or at the very least, require a new analysis to identify potential new safety issues introduced by these new elements.

In the following, a sequence of steps designed to systematically integrate principles from STPA into iStar4Safety is presented. The process is also modeled using Business Process Model and Notation (BPMN) to enhance clarity and support comprehension through visual artifacts. The BPMN representation of the RESafety process is likewise provided.

The level of granularity in each phase of the process depends on the analyst's understanding of the system. In this context, granularity refers to the degree of detail with which the requirements and system elements are specified. For instance, requirements can initially be addressed at a higher level of abstraction—such as general safety goals—and later refined into more detailed elements—such as specific control actions or constraints. Beginning the

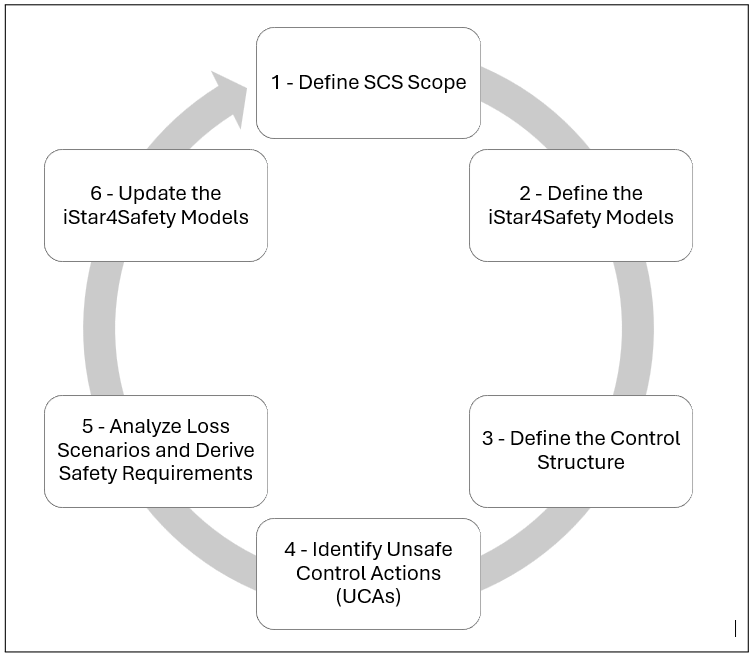
analysis at this higher level may facilitate knowledge management, reducing the risk of omitting essential details when deriving more specific elements from broader concepts (Leveson, 2011). For this purpose, modeling with iStar4Safety—which expresses requirements in the form of actor goals—can be of great assistance. As each iteration contributes to the development of more complete and refined artifacts, the process is characterized as both iterative and incremental.

In this section, which is dedicated to presenting the proposed process, a widely adopted example from the literature—the Insulin Infusion Pump System—is employed. This example was also used in previous studies, including the work that introduced the iStar4Safety modeling language (Ribeiro *et al.*, 2019b; Ribeiro, 2019). The artifact generated during this analysis—specifically, the document detailing the described steps—is available at the following link: ***Access the IIPS analysis***.

3.1.1 Iterative Cycle Diagram

The RESafety process comprises six steps, as illustrated in the diagram in Figure 3.1. As shown, the process begins with Step 1: *Define SCS Scope*, and proceeds sequentially to Step 6: *Update the iStar4Safety Models (if needed)*. The circular arrow connecting the last step back to the first indicates that the process may restart, initiating a new iteration. The steps of the RESafety process are:

Figure 3.1: RESafety Iterative Process



Source: Author (2025).

1. Define the SCS Scope

2. Define the iStar4Safety Models

3. Define the Control Structure
4. Identify Unsafe Control Actions (UCAs)
5. Analyze Loss Scenarios and Derive Safety Requirements
6. Update the iStar4Safety Models (if needed)

Step 1 requests elements that are also required in Step 1 of the STPA analysis. Steps 2 and 6 introduce the possibility of modeling the findings using a GORE language—namely, iStar4Safety. The remaining steps, from Step 3 to Step 5, are grounded in the STPA technique and the artifacts generated during the analysis. The following subsection presents an overview of the BPMN model.

3.1.2 RESafety - Process Overview

Figure 3.2 illustrates all the steps of the RESafety process. The analyst—who may be either a requirements engineer or a safety engineer—initiates the process and proceeds through each of the described steps. At the end of Step 6, a decision must be made as to whether a new iteration will be performed or the process should be concluded, in which case it results in a final safety analysis document.

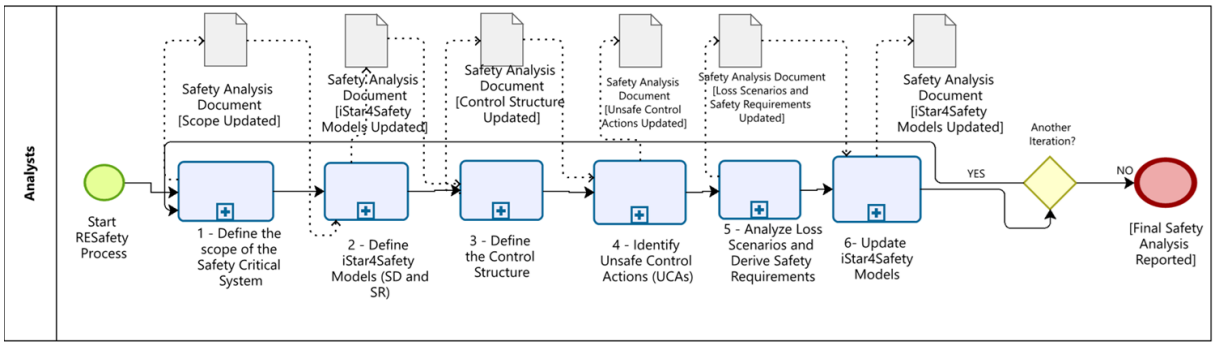
If the process continues to a new iteration, the safety analysis document will be updated at the stage where the iteration resumes. It is important to note, as shown in the BPMN model, that each step generates an updated version of the safety analysis document, reflecting the progress made.

Additionally, the process is designed to be flexible with respect to the order and combination of steps. At the end of this chapter, a section is dedicated to illustrating alternative ways of applying the process, depending on the analyst's decisions.

To access the complete and navigable BPMN model of the RESafety process, visit: ***Access the Complete RESafety BPMN Model***.

Finally, the following subsections provide a detailed explanation of each step, together with an illustrative example based on the IIP System.

Figure 3.2: RESafety Process – Iterative Workflow in BPMN



Source: Author (2025).

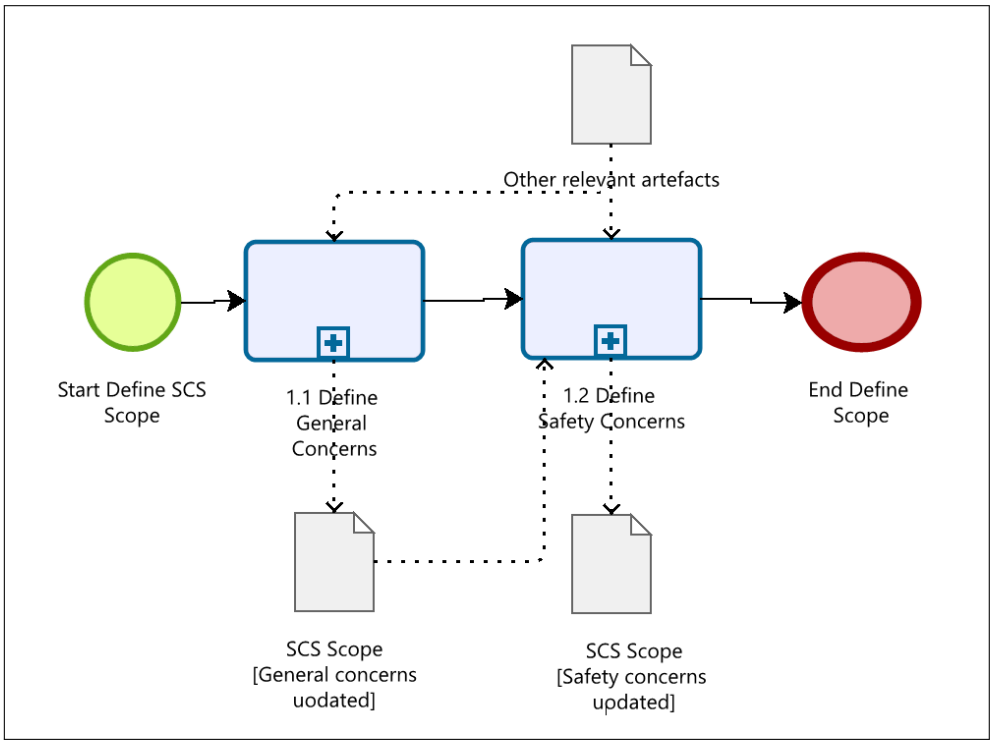
3.1.3 Step 1: Define the scope of the Safety Critical System (SCS)

In the initial phase of the RESafety process, the objective is to define or refine the scope of the system to be modeled. This involves identifying the parts of the system that are relevant for the safety analysis and ensuring that the defined scope aligns with the current stage of development and the available documentation. The output of this phase is an updated Safety Analysis Document [Scope Updated], which serves as the foundation for the subsequent steps.

Given that the process is iterative and incremental, this initial definition may be partial or high-level, particularly during the first iteration. Some elements may be provisionally set or even omitted at this stage. As the process progresses through new cycles, analysts have the opportunity to revisit this scope, refine previously defined aspects, and incorporate new insights or changes arising from the ongoing analysis.

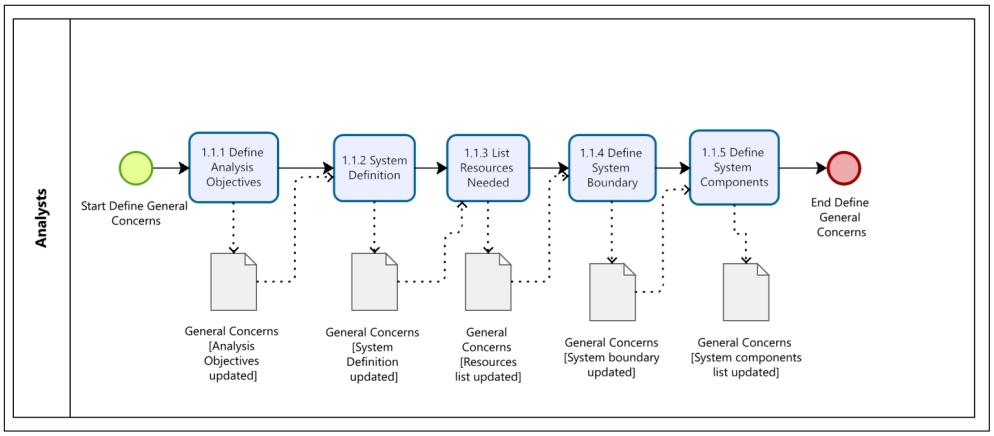
To support this activity, a structured analysis of the elements to be addressed in this step is proposed, categorized into two main groups: General Concerns and Safety Concerns. These categories help ensure a comprehensive and systematic scoping that supports subsequent modeling and analysis efforts. The BPMN model for this step is shown in Figure 3.3, followed by the explanation of each element.

Figure 3.3: Step 1 – Define the scope of the Safety Critical System - BPMN Representation of Step 1 Workflow



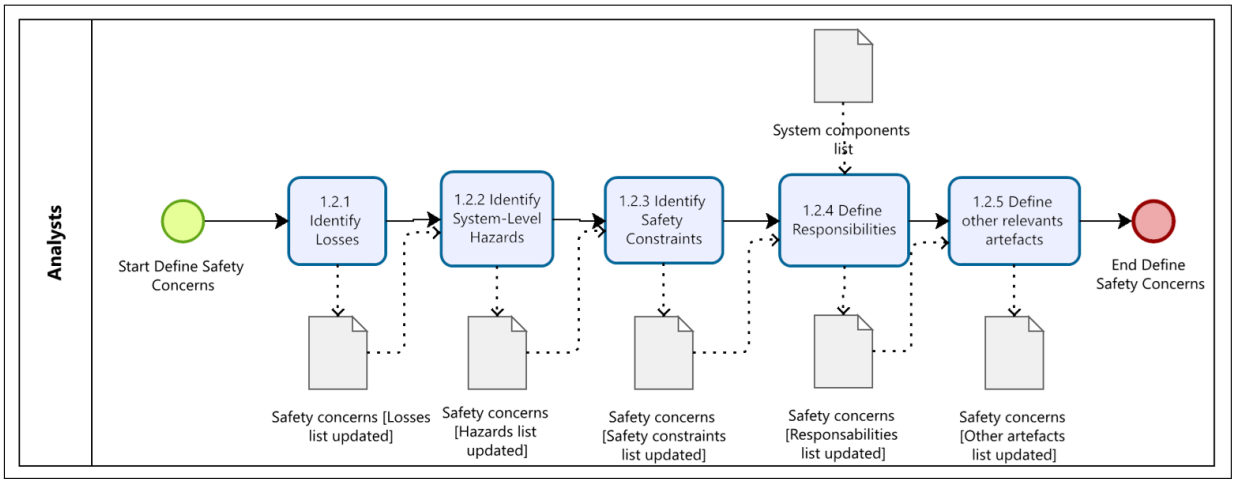
Source: Author (2025).

Figure 3.4: Substep 1.1 – Define general concerns - BPMN Representation of Step 1.1 Workflow



Source: Author (2025).

Figure 3.5: Substep 1.2 – Define safety concerns - BPMN Representation of Step 1.2 Workflow



Source: Author (2025).

- **1.1 - Define General Concerns**: The BPMN model of this subprocess is presented in Figure 3.4.
    - 1.1.1 - Define the analysis objectives: In this activity, the objectives of the analysis must be defined in order to guide the process to be carried out. **IIP System Example:** The purpose of this analysis is to model an Insulin Infusion Pump (IIP) through the iterative RESafety process, generating successive refinements of the system's safety analysis artifacts.
    - 1.1.2 - System Definition: At this stage, the system to be analyzed is briefly introduced, along with a concise explanation of its purpose. **IIP System Example:** The Insulin Infusion Pump (IIP), a safety-critical system, is designed to support the treatment of Type 1 Diabetes Mellitus. Automated IIPs enhance treatment flexibility by managing multiple stages of insulin delivery, effectively mimicking physiological responses. These devices administer both rapid-acting (bolus) and continuous (basal) insulin doses.
    - 1.1.3 - List Resources Needed: At this point, the resources used to conduct the analysis must be listed. These may include system-specific repositories or artifacts, manuals, academic articles, images, prototypes, and other relevant materials. **IIP System Example:** Articles: Martinazzo (2022); Martins *et al.* (2015); Zhang *et al.* (2011, 2010); Bas (2020); Gonzalez Atienza *et al.* (2024), Books Leveson & Thomas (2018); Martins & Gorschek (2021) and other ones, like standards, manuals.
    - 1.1.4 - Define the system boundary: At this stage, the analyst must define the boundaries of the system under analysis. This step is essential for establishing the scope of the analysis, which should ideally focus on the

parts of the system over which control can be exercised to implement safety strategies. **IIP System Example:** The system boundary encompasses activities from the moment the patient configures the infusion settings until the correct dosage is delivered via the catheter.

- 1.1.5 - Define system components: At this stage, the basic components of the system must be identified. These components will be used in system modeling, serving as potential actors in the iStar4Safety models, as well as potential controllers and/or controlled processes in the control structure modeling. **IIP System Example:**
    - Patient
    - Infusion Pump
    - Infusion set

  In subsequent iterations, analysts may refine the identification and specification of system components and subcomponents. Within the IIP system, this may include elements such as control buttons, the LCD/audio interface, the microcontroller, the stepper motor and its driver, the mechanical transmission system, and the insulin syringe.

- **1.2 - Define Safety Concerns**: The BPMN model of this subprocess is presented in Figure 3.5.
    - **1.2.1 - Identify Losses:** A loss refers to any negative outcome that affects something stakeholders consider valuable. Such outcomes may include fatalities or injuries, damage to property, environmental harm, mission failure, reputational damage, leakage of sensitive information, or any other consequence deemed unacceptable by stakeholders (Leveson & Thomas, 2018). It is important to highlight that, according to Leveson & Thomas (2018), in industry practice the terms losses, accidents, mishaps, and even adverse events are often used interchangeably, which can lead to confusion. In this work, the term loss will be used in order to allow a more generalizable concept. In this approach, losses are labeled using the prefix **"Lx"**. References to individual components or specific causes, such as "human error", must be avoided when defining losses (Leveson & Thomas, 2018). **IIP System Example:**
        - L1 - Risk of death
        - L2 - Risk of injury
    - **1.2.2 - Identify System-Level Hazards:** Hazards are system states or set of conditions that in the case of a worst-case environmental condition will

lead to losses (Leveson & Thomas, 2018). When identifying hazards, the analyst must specify the associated losses, ensuring traceability between hazards and losses. Furthermore, suppose the analyst already has UCAs - Unsafe Control Actions, identified from a prior STPA analysis or earlier iterations; verifying whether these UCAs are linked with defined hazards is imperative. Otherwise, analysts must specify a new hazard related to the UCA. This is because UCAs are also the causes of hazards. Avoid referencing individual components in the hazard definition. We can use the **"Hx"** identifier for the hazards. A hazard example, for the loss *"L1 - Risk of death"* is *"H1 - Hypoglycemia [L1, L2]"*. **IIP System Example:**

- H1 - Hypoglycemia [L1, L2]
- H2 - Hyperglycemia [L2]

- **1.2.3 - Identify Safety Constraints.** In this step of the process, the safety constraints for the identified hazards must be defined. It is important to note that, at this point, the focus is not on defining specific solutions or implementations for hazard mitigation, but rather on establishing high-level constraints that must not be violated by the system. These constraints can be formulated as simple negations of the hazards and should be traceable to one or more hazards (Leveson & Thomas, 2018). **IIP System Example:**
  - SC-01: The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
  - SC-02: The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]
- **1.2.4 - Define the responsibilities:** At this point, the analyst must define the responsibilities of each system actor or component to ensure that the established safety constraints are effectively pursued. It is important to note that the components acting as actors in the iStar4Safety models were identified in Step 1.1.5 – Define System Components. These responsibilities represent refinements of the safety constraints and specify what each entity must do to help enforce them. It is recommended that each component be systematically reviewed against every safety constraint, identifying its specific responsibilities—if any—in ensuring that the constraint is enforced. **IIP System Example:** See table 3.1
- **1.2.5 - Define other relevants artefacts:** Finally, the analyst may define relevant artifacts that can be used or are already available and relate to the system's safety. **IIP System Example:** User manual of the infusion pump, provided by Medtronic: ***Access the manual***.

Table 3.1: Component Responsibilities for IIP System

Component	Responsibility
E1 – Patient (Human Controller)	R-01: Ensure that infusion settings are correctly configured and correspond to the medical prescription [SC-01, SC-02]
E2 – Insulin Pump	R-02: Administer insulin only according to validated infusion parameters and prevent unauthorized dosages [SC-01] R-03: Monitor timing and quantity of delivery to ensure correct dose is given at the right time [SC-02] R-04: Detect anomalies (e.g., occlusions, over-delivery) and alert the user immediately [SC-01, SC-02]
E3 – Infusion Set	R-05: Maintain physical integrity to prevent leaks or unintended flow of insulin [SC-01] R-06: Ensure correct and timely delivery of insulin from pump to patient [SC-02]
E4 – Patient (Human Body)	R-07: Respond physiologically to insulin in a way that is consistent with treatment expectations (acknowledging variability) [SC-02]

Source: Author (2025).

It is important to highlight that the actor Patient was divided into two distinct elements: E1 – Patient (Human Controller) and E4 – Patient (Human Body). This decision was inspired by the work of Martinazzo (2022), which modeled the STPA analysis of a real-world insulin pump project. Therefore, this decomposition appeared plausible to us, as it enables a more precise representation of the roles and interactions involved. Specifically, the human can play an active role as the highest-level process in the control structure, while also fulfilling a passive role as the recipient of the treatment.

Finally, documentation, models, safety analyses, and all relevant artifacts associated with the system should be considered as foundational references to support the elicitation of both non-safety and safety requirements. Reusing such materials can help reduce the effort required during the analysis process. These resources may be attached to the analysis itself or referenced and linked to external sources.

3.1.4 Step 2: Define the iStar4Safety Models

In Step 2, RESafety proposes the development of iStar4Safety models. The models — namely, theStrategic Dependency (SD) and the Strategic Rationale (iStar model) / Safety Requirement (RESafety element) (SR) — are used to illustrate the system's early requirements. This high-level perspective provided by iStar4Safety, through the modeling of requirements based on goals and social relationships among actors, can be seen as a support for an analysis that specifies requirements at a high-enough level, improving the management of the analysis.

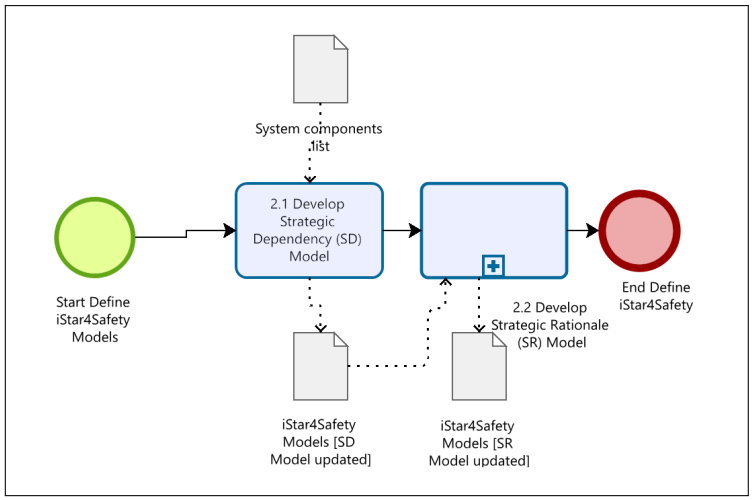
Thus, in Step 2, the actors in the iStar4Safety models will generally correspond to the

components identified in Step 3.1.3 and should be represented accordingly as actors.

Through the SD model, the strategic dependencies among actors are identified, showing who depends on whom and for what (e.g., goals, tasks, resources). This model provides a high-level view of the interactions and responsibilities across system components.

The SR model, in turn, details the internal rationale of each actor, including how they achieve their goals, perform tasks, and manage alternative strategies or conflicting requirements. In this model, the responsibilities defined in Step 1.2.4, as well as the Safety Constraints they aim to address, must be associated with their respective actors. This association supports the justification and traceability of how each safety constraint is considered and enforced within the system's reasoning structure. Figure 3.6 presents the BPM model that illustrates the sequence of steps in this stage.

Figure 3.6: Step 2 - Define iStar4Safety Models (SD and SR) - BPMN Representation of Step 2 Workflow



Source: Author (2025).

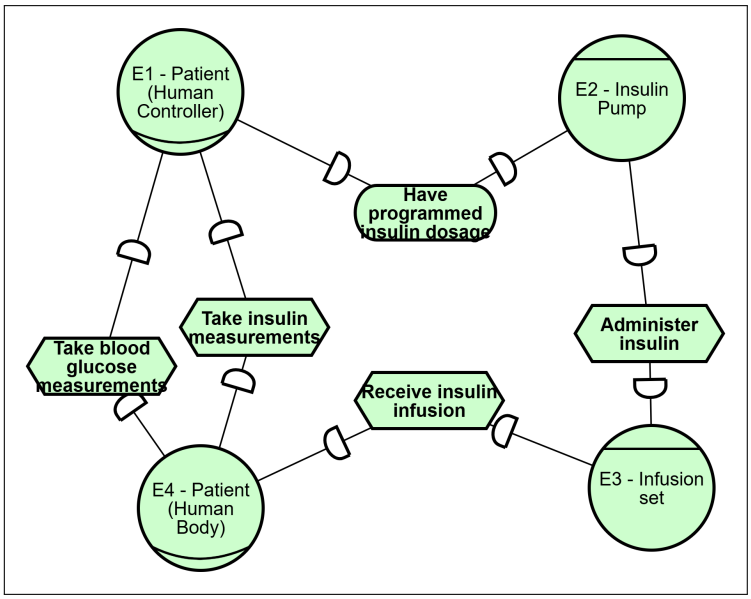
As shown in Figure 3.6, the process begins in Step 2.1 with the development of the SD model and ends in the subprocess of Step 2.2 with the modeling of the SR model. These steps will be further explained below, supported by the IIPS example.

3.1.4.1 SD Model

In Step 2.1 – *Develop Strategic Dependency (SD) Model*, the analyst should have at hand the analysis artifact produced so far—specifically, the list of system components, which is the artifact generated during Step 1.1.5 of Step 1. Based on this, the analyst must analyze and define the actors that will be part of the analysis. Each actor, as well as any element in the analysis, must have an associated ID to support traceability. The actors may be, as defined by iStar4Safety, of type Actor, Agent, or even Role. At this point, for the SD model, the dependencies among the actors should be modeled, which is necessary due to the social interaction nature of iStar4Safety.

It is expected that the actors correspond to those identified in Step 1.1.5. However, the analyst may choose to make relevant adjustments, as will now be illustrated in the IIPS example.

Figure 3.7: IIP System SD Model



Source: Author (2025).

IIP System Example: Figure 3.7 presents the Strategic Dependency (SD) model of iStar4Safety for the IIP System example. Four actors are represented—one more than those defined in Step 1.1.5 of the example. This choice was based on a previous analysis conducted for this pump (Martinazzo, 2022), which was considered an appropriate approach for the early modeling of requirements. The difference lies in the specification of the Patient actor, who may assume both an active role, as a Human Controller – **E1**, and a passive role, representing the human body receiving the treatment – **E4**.

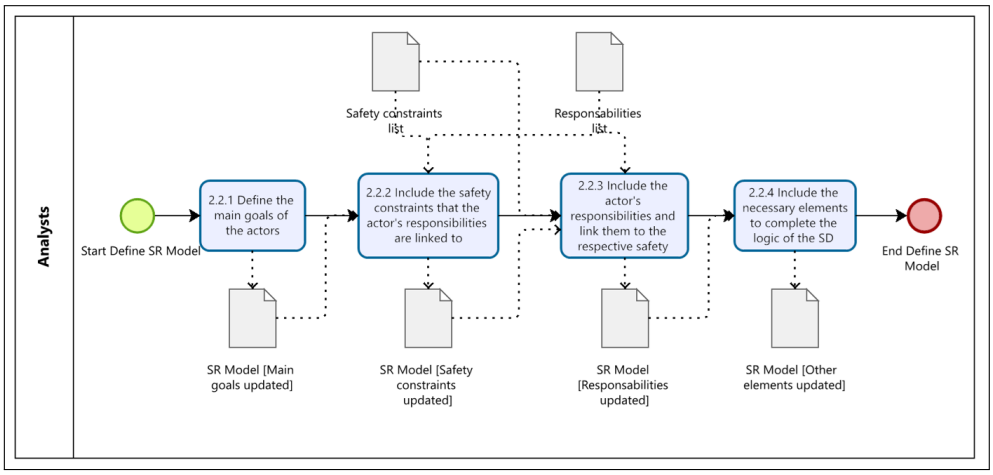
In this model, entity E1 – Patient (Human Controller) depends on entity E2 – Insulin Pump for the "Have programmed insulin dosage" goal. Entity E2 – Insulin Pump, in turn, depends on E3 – Infusion Set to fulfill the "Administer insulin" task. Entity E3 – Infusion Set depends on entity E4 – Patient (Human Body) to perform the "Receive insulin infusion" task. Finally, entity E4 – Patient (Human Body) depends on the patient, now in the role of E1 – Patient (Human Controller), for the "Take blood glucose measurements" and "Take insulin measurements" tasks.

Next, Subprocess 2.2—Develop Strategic Rationale (SR) Model—is described.

3.1.4.2 SR Model

At this stage, the Strategic Rationale (SR) model must be developed. Figure 3.8 depicts the corresponding process in BPM notation.

Figure 3.8: Substep 2.2 – Develop Strategic Rationale (SR) Model -BPMN Representation of Substep 2.2 - Workflow



Source: Author (2025).

In the first iteration, the process begins with the initial SD (Strategic Dependency) model, which, as it already represents the dependencies between actors, may require the inclusion of certain non-safety elements to support this structure.

In Step 2.2.1 – *Define the main goals of the actors*, the overall goal of each actor when interacting with the system must be defined, as the safety logic will be modeled based on this goal. The safety constraints addressed by the actor's responsibilities will be linked to this overall goal.

Therefore, in Step 2.2.2 – *Include the safety constraints that the actor's responsibilities are linked to*, the safety constraints handled by that actor's responsibilities must be modeled as Safety Goals and linked to the actor's main goal. This creates the foundation for modeling all responsibilities associated with that constraint.

In Step 2.2.3 – *Include the actor's responsibilities and link them to the respective safety constraint*, the associated responsibilities defined in Step 1.2.4 of the Safety Concerns are modeled as Safety Goals and connected to the Safety Constraints they are meant to fulfill. These associations must be based on the artifacts *Safety Constraints [Safety Constraints List Updated]* and *Safety Constraints [Responsibilities List Updated]*, generated in Steps 1.2.3 and 1.2.4, respectively.

Finally, in Step 2.2.4 – *Include the necessary elements to complete the logic of the SD*, the analyst may model any remaining dependency elements whose internal logic and fulfillment fall within the scope of that specific actor. After completing all these steps, the SR model is finalized and included in the safety analysis document.

It is important to emphasize that all IDs and associated elements must always be modeled to support traceability among components. All responsibilities and their associated safety constraints should be modeled. The Safety Constraints (SC) element, therefore, should be linked to the actor's overall goal—already modeled—as a Safety Goal. The Responsabilities (R)

associated with that SC must also be connected to it and modeled as Safety Goals.

This modeling approach reflects the rationale that responsibilities are meant to show how each actor contributes to fulfilling the SR, which in turn is a restriction designed to mitigate the occurrence of System-Level Hazards (H) defined in Step 1.2.2. If a R is not performed, the corresponding SR may not be satisfied, which could lead to the occurrence of the associated H modeled in Step 1.2.2—ultimately resulting in the Accidents (A) described in Step 1.2.1.

IIP System Example: The example presented here, shown in Figure 3.9, corresponds to each actor defined in the SD model, now with their boundaries also defined. Due to space constraints, and given that the modeling of the other actors can be understood based on this case, only actor E1 – Patient (Human Controller) is detailed as a representative example.

The main goal of E1 actor is "Use the IIP". Safety constraints SC-01 and SC-02 were added to this goal, since the actor hold responsibility R-01, which aim to address those specific safety constraints. The associations between R-01 and safety constraints SC-01 and SC-02, are indicated at the actor's detailed modeling.

Finally, during the analysis, the need was identified to model additional non-safety elements previously included in the SD model: namely, the goal "Have programmed insulin dosage", fulfilled by actor E2, and the tasks "Measure blood glucose" and "Measure insulin", fulfilled by actor E1 due to its dependency relationship with actor E4.

In the following section, Step 3 is presented, focusing on the modeling of the system's control structure.

3.1.5 Step 3: Define the Control Structure

Figure 3.10 presents the BPMN model of Step 3 of the RESafety process, which is dedicated to defining the control structure of the system under analysis. This step is essential to support the identification of UCAs, HCs and safety requirements in subsequent steps. It offers flexibility to the analyst by distinguishing between mandatory and optional modeling elements, depending on the system's complexity and safety relevance.

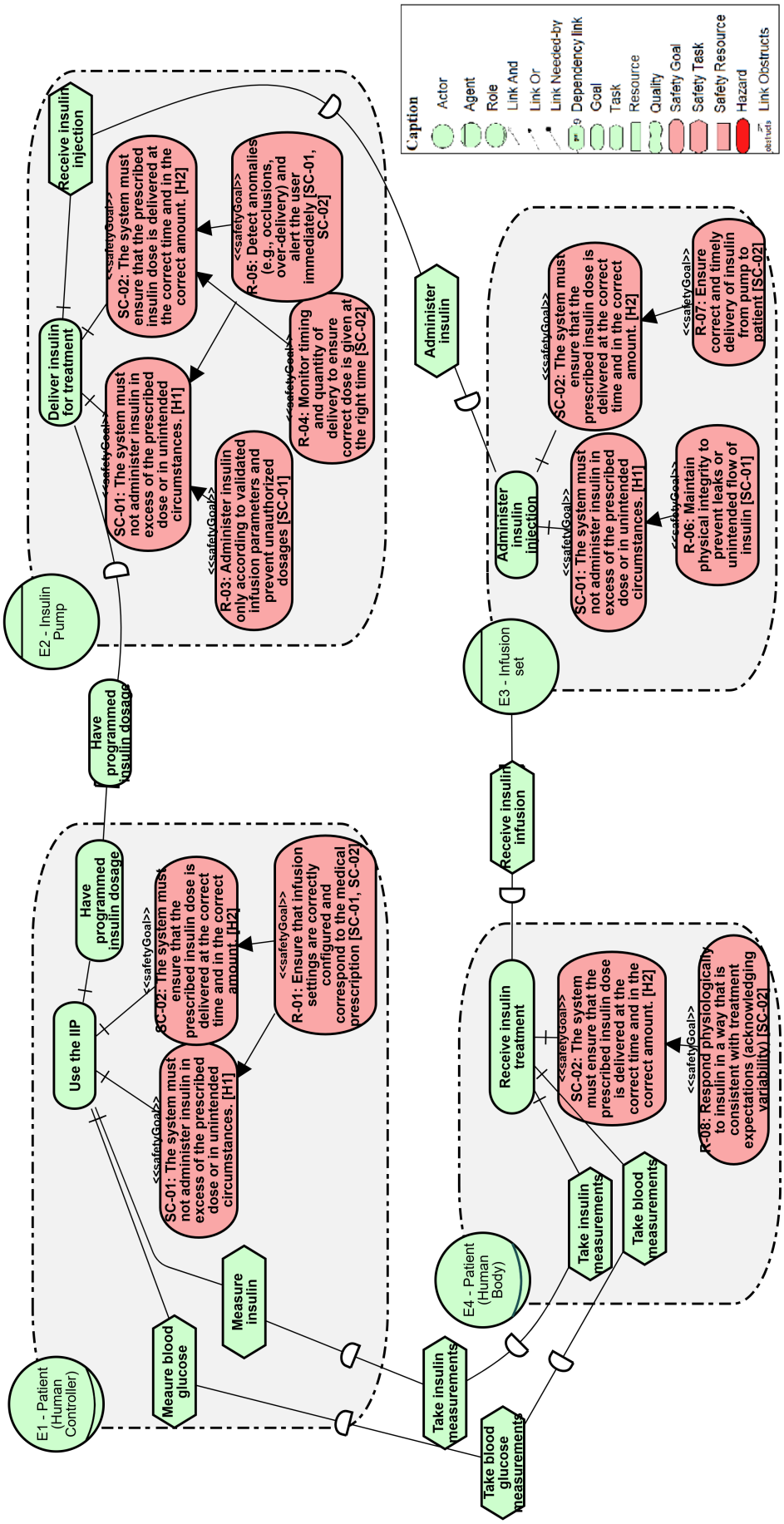
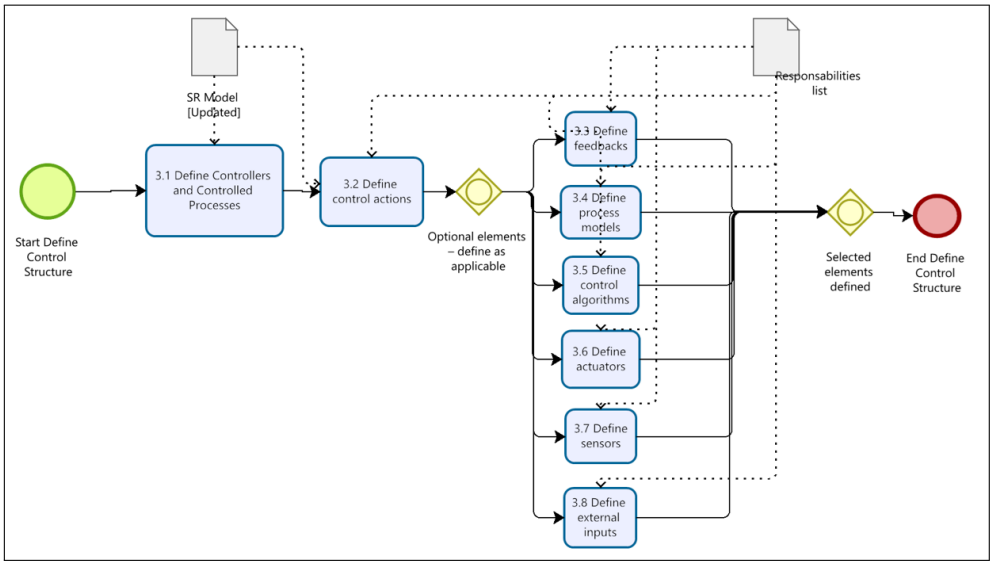


Figure 3.9: IIP System SR Model.

Source: Author (2025).

Figure 3.10: Step 3 - Define the Control Structure - BPMN Representation of Step 3 Workflow



Source: Author (2025).

The subprocess begins with Step 3.1 – *Define Controllers and Controlled Processes*. In this task, the analyst must define the entities that act as *controllers*—components responsible for decision-making and issuing commands—and those that serve as *controlled processes*—components that react to commands. It is important to note that a single process may simultaneously act as a controller for certain elements while being controlled by others. This definition should be guided by the actors identified in the updated Strategic Rationale (SR) model.

The next step is 3.2 - *Define Control Actions*, where the analyst must identify the control actions issued by the controllers to the controlled processes. These actions are often derived from the responsibilities assigned to each actor in the iStar4Safety model and are fundamental for Step 4, where they are analyzed as potential Unsafe Control Actions (UCAs). This task is mandatory because the process must define control actions to proceed with the UCA analysis.

To model the control actions, the analyst should use the SD model defined in Step 2.1 as a reference. Based on this model, control actions must be derived for each relevant entity. Specifically, in any dependency relationship, the depender (the actor that initiates the dependency) is considered the controller, while the dependee (the actor that fulfills the dependency) is the controlled process. This interpretation stems from the fact that the depender requests something from the dependee; therefore, the dependum of the relationship is modeled as the corresponding control action.

Once these control actions are identified, they should be linked to the responsibilities they support or implement.

An alternative modeling scenario arises when a passive entity is involved—one that does not initiate any control actions and is only influenced by other components. In such cases, this entity should be modeled exclusively as a controlled process. The dependum of its associated

dependency should then be represented as feedback from the controlled process to the controller, reflecting the flow of information rather than a command.

In other words, it is not always possible to associate responsibilities directly with control actions. However, doing so—when applicable—can be beneficial to the analysis. Control actions, when evaluated as potential Unsafe Control Actions (UCAs) in Step 4, enable a more structured and detailed safety assessment than responsibilities addressed solely through other control structure elements.

Nonetheless, the original STPA reference (Leveson & Thomas, 2018) does not require that every responsibility be linked to at least one control action. Responsibilities may be addressed indirectly through components such as feedback loops, process models, or control algorithms, as previously mentioned.

When a responsibility is not directly associated with a control action, it may not lead to the identification of a UCA, and its safety logic might therefore not follow the conventional UCA-to-safety-requirement modeling path. Nonetheless, the absence of an associated control action does not imply that the responsibility is disregarded. In the context of STPA, a hazard is defined as a system state or condition that, when combined with certain environmental or operational circumstances, may lead to a loss (Leveson, 2011). Hazards represent high-level unsafe conditions that can emerge from the behavior of the system as a whole. To complement this perspective, RESafety introduces the concept of Hazardous Conditions (HCs), defined as potential hazardous situations not captured by UCAs but still critical to system safety. HCs provide a mechanism to represent unsafe situations that may arise, for example, from the non-fulfillment of responsibilities or from specific contextual factors. Thus, while hazards constitute the traditional focus of system-theoretic safety analysis, HCs extend the scope by ensuring that relevant unsafe situations not covered by UCAs are also systematically addressed. Analysts are therefore encouraged to introduce HCs when necessary (Ribeiro *et al.*, 2024).

Furthermore, it is important to consider the influence of external inputs—elements outside the system boundary that can still affect its behavior. Although these inputs are not part of the system itself, they may play a crucial role in the fulfillment of responsibilities. For example, a medical prescription issued by a physician (an external actor) directly impacts the responsibility of ensuring that the infusion settings are correctly configured. Therefore, even inputs that fall outside the direct control scope must be properly identified and modeled to support a complete and accurate safety analysis.

Following the two initial steps, the process reaches a **gateway** labeled *Optional elements – define as applicable*. From this point, the analyst may model additional control structure elements based on what is relevant for the system and the depth of understanding already available. The more high-level and abstract the modeling at this stage, is the greater the potential to prevent information loss in later specification phases. However, these optional elements can significantly improve the depth and completeness of the safety analysis.

The optional tasks are as follows:

- **3.3 Define Feedbacks** – Identify and model feedback mechanisms through which controllers receive information about the state of the controlled processes.
- **3.4 Define Process Models** – Specify the internal models used by controllers to understand the process state. These are essential for detecting mismatches between expected and actual behavior.
- **3.5 Define Control Algorithms** – Define the decision-making logic used by controllers to generate control actions. This may involve rules, logic-based routines, or more complex algorithmic behavior.
- **3.6 Define Actuators** – Identify the components responsible for physically or logically implementing the control actions.
- **3.7 Define Sensors** – Define the components that collect process data and transmit it back to controllers, often forming feedback loops.
- **3.8 Define External Inputs** – Model external influences such as user commands, environmental conditions, or other inputs that affect system behavior and safety.

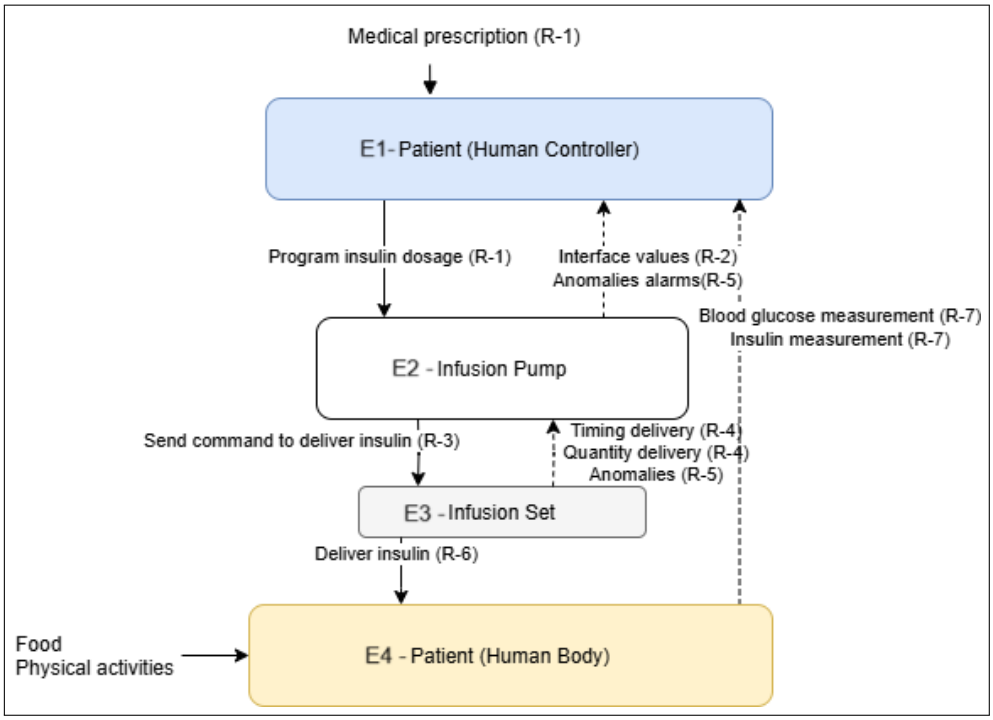
Each of these optional modeling tasks is supported by references to the SR model and the responsibilities list (as indicated by the dashed arrows in the diagram), ensuring consistency and traceability across the analysis.

After the selected elements have been modeled, the subprocess is concluded, resulting in an updated control structure as its main artifact.

It is important to emphasize that this structure makes Step 3 adaptable to the system's design stage and specific project needs. While only Steps 3.1 and 3.2 are mandatory, modeling additional elements can offer greater insight and enhance the robustness of the safety analysis. This flexibility is particularly relevant in early design phases, and the iterative nature of RESafety is intended to address this need by allowing progressive refinement of the control structure throughout successive iterations.

IIP System Example: For the IIP System, considering the previous steps and their inputs, the Control Structure shown in Figure 3.11 was generated. It should be noted that the responsibilities were associated with the elements intended to fulfill them to ensure traceability. We highlight the importance of explicitly associating each responsibility with the elements that address it, in order to support traceability and ensure consistency throughout the modeling process.

Figure 3.11: Control Structure Model of the IIP System



Source: Author (2025).

To model control actions and feedback, the SD model generated in Step 2.1, presented in Figure 3.7, was used as a reference. Table 3.2 presents the associations between elements from the iStar4Safety model and their corresponding elements in the control structure.

Table 3.2: Mapping Between iStar4Safety SD Elements and Control Structure Elements

iStar4Safety SD Element	**Representation in iStar4Safety**	**Control Structure Element**	**Representation in Control Structure**
Have programmed insulin dosage	Goal Dependency – E1 to E2	Program insulin dosage	Control Action – E1 to E2
Administer insulin	Task Dependency – E2 to E3	Send command to deliver insulin	Control Action – E2 to E3
Receive insulin infusion	Task Dependency – E3 to E4	Deliver insulin	Control Action – E3 to E4
Take blood glucose measurement	Task Dependency – E4 to E1	Blood glucose measurement	Feedback – E4 to E1
Take insulin measurement	Task Dependency – E4 to E1	Insulin measurement	Feedback – E4 to E1

Source: Author (2025).

As shown in Figure 3.11, Responsibility R-1 — "Ensure that infusion settings are correctly configured and correspond to the medical prescription [SC-01, SC-02]" — is addressed through the control action *"Program Insulin Dosage (R-1)"*, executed by process E1 over entity E2, and also through the external input *"Medical Prescription (R-1)"*.

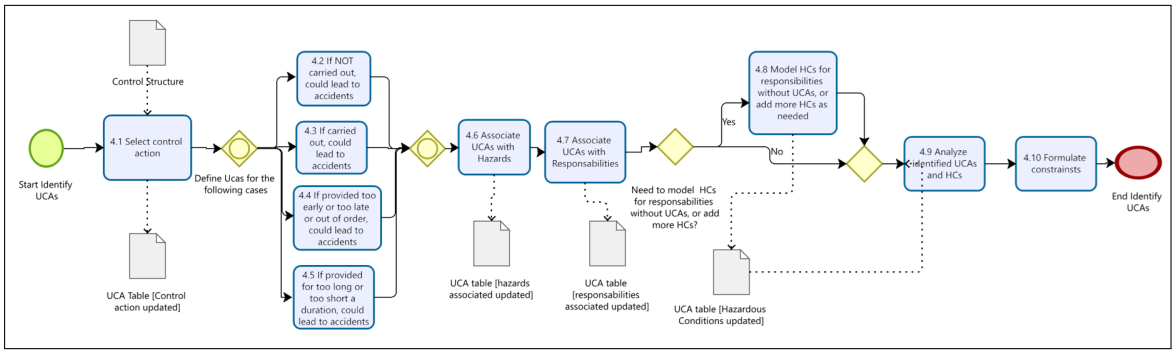
The remaining responsibilities were addressed through control actions and feedback mechanisms accordingly.

It is also important to highlight that the controlled process E4 — *Patient (Human Body)* — due to its passive role in the treatment and inability to perform control actions, had its tasks modeled as feedback. This modeling approach effectively captured the nature of the system and contributed to a coherent representation.

3.1.6 Step 4: Identify Unsafe Control Actions (UCAs)

The BPMN model presented in Figure 3.12 visually represents the subprocess dedicated to identifying Unsafe Control Actions (UCAs), corresponding to Step 4 of the RESafety process. Rooted in the STPA methodology, this step aims to systematically identify control actions that may result in hazardous behavior under specific circumstances. In addition, the identification of Controller Constraints must also be performed, as illustrated in the BPMN model and further detailed in Steps 4.9 and 4.10.

Figure 3.12: Step 4 - Identify Unsafe Control Actions (UCAs) - BPMN Representation of Step 4 Workflow



Source: Author (2025).

With the control structure available, the analyst initiates the subprocess by selecting a control action to be evaluated (Step 4.1). According to Leveson and Thomas (Leveson & Thomas, 2018), four specific situations must be assessed for each control action to determine if it could be unsafe:

- If the control action is **not carried out**, it could lead to losses;
- If the control action is **carried out**, it could lead to losses;
- If the control action is provided **too early, too late, or out of order**, it could lead to losses;
- If the control action is provided for **too long or too short a duration**, it could lead to losses.

Each of these conditions is evaluated individually. When applicable, they give rise to one or more UCAs (Steps 4.2 to 4.5), which are recorded in a UCA table associated with the control action under analysis.

Following this, in Step 4.6, each UCA must be linked to one or more system-level hazards identified earlier in Step 1.1.2. This association is essential for ensuring traceability between potential unsafe behaviors and high-level system hazards. The RESafety approach refines this process by also associating each UCA with the responsibility and corresponding SC that govern its origin (Step 4.7).

To support this traceability, analysts are encouraged to follow this structured reasoning path:

1. Identify the responsibility associated with the Unsafe Control Action (UCA), which should be indicated in the control action that originated it;
2. Determine the Safety Constraint associated with that responsibility;
3. Trace the Safety Constraint to the corresponding system-level Hazard.

This step reinforces the link between behavioral safety issues and the functional responsibilities modeled earlier in the process.

It is important to note that some responsibilities may not result in UCAs, particularly when fulfilled through passive feedback mechanisms or external inputs. As discussed in Section 3.1.5, for instance, Responsibility R-06 — "Ensure correct and timely delivery of insulin from the pump to the patient [SC-02]" — is fulfilled exclusively through feedback (i.e., *Blood glucose measurement (R-6)* and *Insulin measurement (R-6)*) sent from E4 – Patient (Human Body) to E1 – Patient (Human Controller). Since no control action is involved, no UCA is generated. However, the safety impact of this responsibility still needs to be considered.

To address such cases, Step 4.7 includes the task "Associate UCAs with Responsibilities." When a UCA cannot be linked to any responsibility—impacting the completeness of the safety model—RESafety introduces the concept of Hazardous Conditions (HC). These represent potential hazardous situations not captured by UCAs but still critical to system safety. Step 4.8 then allows the analyst to determine whether such HCs should be modeled.

This decision point—*"Need to model HCs for responsibilities without UCAs, or add more HCs?"*—is evaluated under two possible conditions:

- **Yes:** Proceed to Step 4.8, which enables:
  - Modeling HCs for responsibilities not linked to any UCAs due to the absence of associated control actions;
  - Modeling additional hazards not covered by existing UCAs, even if the responsibility already has an associated control action.

- **No:** When no additional HCs are required, the process directly advances to Steps 4.9 and 4.10, where the previously identified UCAs and existing HCs are systematically analyzed to derive the Controller Constraints.

At this point, the analyst transitions from the diagnostic perspective—focused on identifying potential unsafe control actions and hazardous conditions - to the prescriptive perspective, aimed at defining how the controllers must behave to maintain system safety.

Step 4.9: Analyze Identified UCAs and HCs consists of a detailed review of the UCA and HC tables generated in the earlier tasks. In this activity, the analyst examines each unsafe control action and hazardous condition to determine the specific behavioral or operational context that led to its identification. This analysis ensures that the causal relationships between control actions, hazards, and responsibilities are traceable and logically consistent.

Step 4.10: Formulate Controller Constraints operationalizes the safety reasoning developed in previous steps. Each analyzed UCA or HC is translated into one or more explicit Controller Constraints, which describe how the system's controllers must act—or what they must avoid doing—to prevent unsafe behaviors or hazardous situations. These constraints serve as the formal link between the hazard analysis and the design phase, establishing verifiable safety conditions that can be implemented, monitored, or validated.

In summary, the BPMN model in Figure 3.12 outlines a structured, traceable, and iterative approach for identifying Unsafe Control Actions that integrates system-level hazards, responsibilities, and safety constraints. This approach ensures that:

- All relevant UCAs are systematically identified and evaluated;
- Each UCA is linked to at least one system-level hazard and, when applicable, to one or more responsibilities and safety constraints;
- All responsibilities are addressed—either through UCAs or through associated HC;
- Each UCA and HC results in one or more Controller Constraints, ensuring their translation into actionable safety requirements;
- The safety model evolves through iterative refinement, maintaining consistency across hazards, responsibilities, UCAs, HCs, and constraints.

This integration of behavioral and structural elements within the RESafety process supports a more complete and robust safety assessment.

IIP System Example: For the IIP System now apresentadmos in table 3.3 an example of how a control action can result in each of the four types of Unsafe Control Actions (UCAs). The selected control action is: "Program insulin dosage (R-1)".

As shown in the table above, it exemplifies the analysis of UCA for a specific control action defined in the control structure: "Program insulin dosage (R-1)". This action is performed

Table 3.3: Unsafe Control Actions (UCAs) for the control action “Program insulin dosage (R-1)”

Control Action	From/To	Not Providing Causes Hazard	Providing Causes Hazard	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Program insulin dosage (R-1)	Patient / Infusion Pump	**UCA-01:** Patient does not provide “Program insulin dosage” when insulin is required, leading to underdose [H1]	**UCA-02:** Patient provides “Program insulin dosage” with a value higher than prescribed, leading to overdose [H2] **UCA-03:** Patient provides “Program insulin dosage” with a value lower than prescribed, leading to underdose [H1]	**UCA-04:** Patient provides “Program insulin dosage” too late, leading to hyperglycemia [H1] **UCA-05:** Patient provides “Program insulin dosage” too early, leading to premature insulin administration and resulting in hypoglycemia [H2]	Not applicable

Source: The author (2025).

by E1 – *Patient (Human Controller)* and directed to E2 – *Infusion Pump*. It is associated with Responsibility R-1 and aims to ensure that the insulin dosage delivered by the pump is consistent with the prescribed medical treatment.

For each control action, the RESafety method evaluates its potential to become unsafe under the four STPA-defined conditions. In this case:

- **UCA-01:** If the patient fails to program the infusion pump (i.e., the control action is not provided), the pump will not deliver insulin as needed, potentially leading to **Hazard H1**, which corresponds to the risk of hypoglycemia that may result in serious injury or even death.
- **UCA-02:** If the patient programs an insulin dosage higher than prescribed, this may lead to an overdose and trigger **Hazard H2**.
- **UCA-03:** If the patient programs an insulin dosage lower than prescribed, this may lead to insufficient treatment and trigger **Hazard H1**.
- **UCA-04:** If the patient programs the insulin too late—after the appropriate time window—this may result in hyperglycemia, thus leading to **Hazard H1**.
- **UCA-05:** If the patient programs the insulin too early—before the required time—this may result in hypoglycemia, thus leading to **Hazard H2**.
- The column corresponding to **duration-related issues** (too long or too short) is marked as *Not applicable* in this scenario, indicating that the duration of this control action is not considered a relevant safety concern in this context.

Furthermore, during the decision gateway of the analysis process, the analysts decided to proceed to Step 4.8, as they identified the need to add a new Hazardous Condition (HC)—that is, a potentially dangerous situation not previously covered by the defined UCAs. In this case, the specific condition added was **HC-01: The pump is misplaced or inaccessible to the patient**.

This decision illustrates a key point in the RESafety process: the need to account for responsibilities not addressed by any UCA. In such cases, modeling additional HC ensures comprehensive safety coverage. Additionally, it was determined that in the next iteration, a further analysis would be conducted to assess whether this HC should be linked to an existing system-level hazard or if a new hazard must be defined. This follows the RESafety guideline that every UCA or HC must be associated with at least one hazard. If no appropriate hazard is currently available in the model, a new one must be created.

In the RESafety process, the identification of UCA and HC is directly followed by the definition of the corresponding Controller Constraints. Therefore, the activity previously represented as Step 5 is here integrated into Step 4, maintaining the same logical continuity of analysis.

Once the analyst has completed the identification of UCAs and, when necessary, HCs, the next part of Step 4 focuses on transforming these findings into explicit Controller Constraints. These constraints specify how each controller must behave—or what it must avoid doing—in order to prevent the occurrence of unsafe control actions or hazardous conditions. This integration ensures a seamless transition between the analytical identification of unsafe behavior and the prescriptive definition of safety rules.

In the Step 4 the identification of UCA and HC is directly followed bythe definition of the corresponding Controller Constraints. T

By analyzing these artifacts, the analyst derives the specific conditions under which the system must behave to ensure safety. These conditions are then formally translated into constraints during task 4.9- **Formulate Constraints**. Each constraint specifies how a controller must behave—or what it must avoid doing—in order to prevent the associated hazards from materializing.

The result of this substep within Step 4 is a list of **Controller Constraints**, which is updated and stored to support subsequent steps in the RESafety process.

Overall, this integration operationalizes safety by ensuring that every UCA and HC leads to the explicit definition of a constraint that can be implemented or verified in the system's design or runtime behavior. It also ensures traceability by linking each constraint back to the specific hazard scenario or unsafe behavior it addresses.

IIP System Example: Table 3.4 presents the Controller Constraints derived from the analysis of each UCA identified for the IIP System, which has been used as an illustrative example throughout this process. Specifically, the constraints correspond to the UCAs and HC identified for the control action *"Program insulin dosage (R-1)"*.

In conclusion, Step 4 now fully encompasses both the identification of Unsafe Control Actions and the definition of the corresponding Controller Constraints. This unified structure provides a continuous analytical and prescriptive flow, ensuring traceability from hazards to responsibilities, UCAs, HCs, and finally to safety-enforcing constraints that can be implemented or verified during system design and operation.

3.1.7 Step 5: Analyze Loss Scenarios and Derive Safety Requirements

In Step 5, the analyst is responsible for examining the Loss Scenario (LS)s, which correspond to the causal factors that lead to the occurrence of UCA and or HC, and consequently, to system hazards. This step can be demanding, given its typically ad hoc modeling nature. In the STPA Handbook (Leveson & Thomas, 2018), the authors outline a set of procedures for identifying such scenarios. Furthermore, a video-based presentation[1] was developed by the authors to introduce a more systematic approach to the identification of loss scenarios.

[1]Video titled STPA: Formally Developing Loss Scenarios, available at `https://www.youtube.com/watch?v=hp-KBjIBmrI&t=3936s`

Table 3.4: Controller Constraints for “Program insulin dosage” action

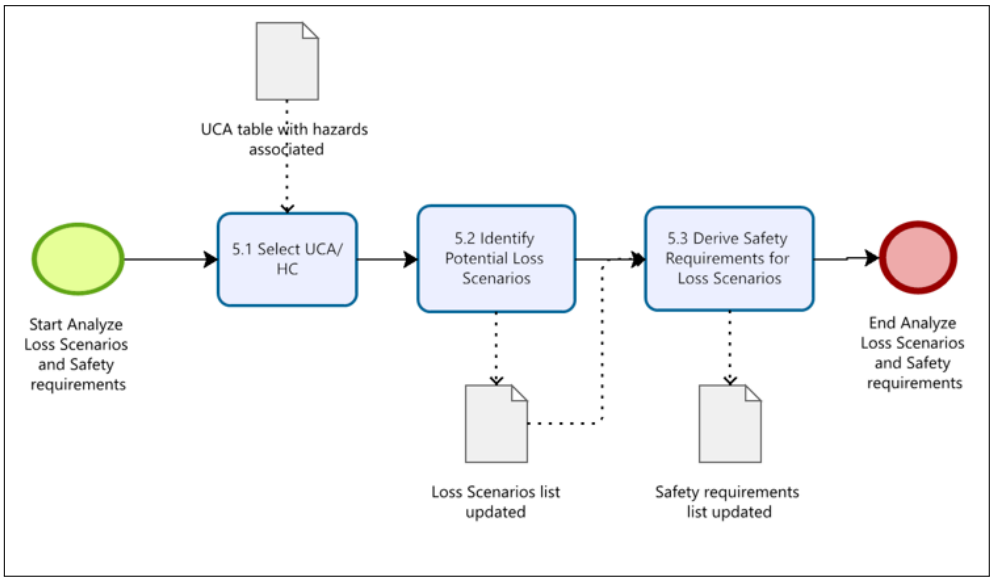
UCA/HC	Controller Constraint
UCA-01: Patient does not provide “Program insulin dosage” when insulin is required, leading to underdose [H1]	**C-01:** The patient must program the insulin dosage whenever insulin is required, according to clinical guidance. [UCA-01]
UCA-02: Patient provides “Program insulin dosage” with a value higher than prescribed, leading to overdose [H2]	**C-02:** The patient must ensure the programmed insulin dosage does not exceed the value prescribed by the physician. [UCA-02]
UCA-03: Patient provides “Program insulin dosage” with a value lower than prescribed, leading to underdose [H1]	**C-03:** The patient must verify that the programmed dosage meets the minimum prescribed threshold to avoid underdosing. [UCA-03]
UCA-04: Patient provides “Program insulin dosage” too late, leading to hyperglycemia [H1]	**C-04:** The patient must program the insulin dosage in a timely manner, according to the prescribed administration window. [UCA-04]
UCA-05: Patient provides “Program insulin dosage” too early, leading to premature insulin administration and resulting in hypoglycemia [H2]	**C-05:** The patient must not program the insulin dosage before the appropriate physiological or dietary condition occurs. [UCA-05]
HC-01: The pump is misplaced or inaccessible to the patient.	**C-06:** The pump must be located in an accessible and known location to the patient at all times. [HC-01]

Source: The author (2025).

Regardless of the method adopted for identifying loss scenarios, the main objective of this step is to uncover them and define corresponding mitigation strategies. These mitigation strategies take the form of safety requirements. To support traceability, all elements—UCAs, loss scenarios, and safety requirements—should be explicitly linked. As highlighted by Leveson & Thomas (2018), the outputs generated during an STPA analysis can serve multiple purposes. These include, but are not limited to, the elicitation of safety requirements, the identification of design recommendations, and the refinement or guidance of the system’s architecture.

Figure 3.13 illustrates, using BPMN notation, the subprocess corresponding to Step 5 of the RESafety process.

Figure 3.13: Step 5 - Analyze loss scenarious and derive safety requirements - BPMN Representation of Step 5 Workflow



Source: Author (2025).

The subprocess begins with the Step **5.1 Select UCA/HC**, where the analyst selects one item at a time from the list of UCAs or HCs (introduced in Step 4). These elements are the entry point for the loss-based reasoning that characterizes this step.

In **Step 5.2 – Identify Potential Loss Scenarios**, the analyst investigates specific combinations of conditions, environmental factors, or operational errors that could cause the selected UCA or HC to result in a system-level hazard. Each potential path leading from the UCA/HC to a hazard is registered as a unique Loss Scenario. These are documented in an evolving list that supports traceability.

Next, in **Step 5.3 – Derive Safety Requirements for Loss Scenarios**, the analyst formulates safety requirements designed to prevent or mitigate the loss scenarios identified. Each SR is derived with the objective of eliminating the causal factors or conditions that would allow the loss to occur. These SRs are recorded in a dedicated artifact and become integral to the overall safety specification of the system.

Throughout this process, traceability is maintained:

- Each Loss Scenario is linked back to one or more UCAs or HCs.
- Each Safety Requirement is explicitly linked to one or more Loss Scenarios.

In summary, Step 5 of the RESafety process supports:

- Exploration of how unsafe behaviors could manifest in practice;
- Formulation of safety requirements tailored to prevent or mitigate those manifestations;

- Traceability between UCAs/HCs, loss scenarios, and associated system-level safety requirements.

IIP System Example: The table 3.5 present the application of Step 5 of the RESafety process to the IIP system. In this step, each identified UCA and HC is analyzed to derive potential LS, which are then used to formulate system-level Safety Requirements (SR).

Table 3.5: Loss Scenarios and Safety Requirements for IIP System

UCA/HC	Loss Scenario (LS)	Safety Requirement (SR)
UCA-01: Patient does not provide "Program insulin dosage" when insulin is required, leading to underdose [H1]	**LS-01:** The patient forgets to program the dose after the meal, resulting in hyperglycemia. [UCA-01] Martinazzo (2022) **LS-02:** The system does not issue a reminder to program the dose after detecting a meal event. [UCA-01] Ribeiro *et al.* (2024)	**SR-01:** The system shall generate an alert if insulin is not programmed within 15 minutes after a meal is detected. [LS-01] Zhang *et al.* (2011) **SR-02:** The interface must maintain a visible warning if no insulin programming is detected post-meal. [LS-02] Ribeiro *et al.* (2024)
UCA-02: Patient provides "Program insulin dosage" with a value higher than prescribed, leading to overdose [H2]	**LS-03:** The patient repeats a bolus due to lack of feedback on recent insulin administration. [UCA-02] Zhang *et al.* (2010) **LS-04:** Patient misinterprets the prescribed dose and enters a value higher than medically indicated. [UCA-02] Zhang *et al.* (2011)	**SR-03:** The system shall display recent insulin activity clearly before accepting a new dose. [LS-03] Zhang *et al.* (2011) **SR-04:** The system shall cross-check manual input with prescription data and alert if excess dosage is detected. [LS-04] Zhang *et al.* (2011)
UCA-03: Patient provides "Program insulin dosage" with a value lower than prescribed, leading to underdose [H1]	**LS-05:** The patient reduces the dose to avoid hypoglycemia without clinical basis. [UCA-03] Martinazzo (2022) **LS-06:** The system does not notify that the entered dose is below clinical expectation. [UCA-03] Zhang *et al.* (2011)	**SR-05:** The system must recommend confirmation when the user's dose is significantly below the recommended amount. [LS-05, LS-06] Zhang *et al.* (2011)
UCA-04: Patient provides "Program insulin dosage" too late, leading to hyperglycemia [H1]	**LS-07:** The patient delays programming due to being busy or distracted, compromising glycemic control. [UCA-04] Martinazzo (2022) **LS-08:** The system accepts bolus entry even after blood glucose spike already occurred. [UCA-04] Ribeiro *et al.* (2024)	**SR-06:** The interface must issue periodic prompts for pending bolus if blood glucose remains elevated and no dose is scheduled. [LS-07] Zhang *et al.* (2011) **SR-07:** The system must block bolus entries considered ineffective post-prandial, requiring physician override. [LS-08] Ribeiro *et al.* (2024)
UCA-05: Patient provides "Program insulin dosage" too early, leading to premature insulin administration and resulting in hypoglycemia [H2]	**LS-09:** Patient programs insulin and forgets to eat, leading to insulin drop without carbohydrate intake. [UCA-05] Zhang *et al.* (2010) **LS-10:** Patient assumes a meal is imminent, but it is delayed due to unforeseen events. [UCA-05] Martins *et al.* (2015)	**SR-08:** System must require user confirmation that the meal is occurring before completing bolus delivery. [LS-09] Zhang *et al.* (2011) **SR-09:** If a meal confirmation is not received within a set time, bolus must be suspended or canceled automatically. [LS-10] Martins *et al.* (2015)
HC-01: The pump is misplaced or inaccessible to the patient.	**LS-11:** The patient is in a critical condition and does not remember where the pump was placed.	**SR-10:** The pump must have an associated mobile application that allows a "locate pump" function to trigger an audible alarm when activated.

Source: The author (2025).

As shown, UCAs such as "UCA-01: Patient does not provide 'Program insulin dosage' when insulin is required leading to underdose" lead to specific loss scenarios (e.g., LS-01 and LS-02), both associated with the hazard of hypoglycemia (H1). Each loss scenario is documented with its respective source or rationale. For example, *LS-01* considers a situation where the patient forgets to program the dose after a meal, potentially leading to hyperglycemia, and *SR-01* is derived to require an alert if insulin is not programmed within 15 minutes of a meal.

The analysis continues for other UCAs (UCA-02 to UCA-05) and the HC-01, each mapped to plausible hazardous situations and resulting in concrete safety requirements.

Importantly, this example reinforces the traceability between UCAs/HCs, LSs, and SRs, which is a core principle of RESafety. Each requirement generated is explicitly linked back to one or more loss scenarios and their originating unsafe actions or HCs. This ensures justified set of safety requirements, directly addressing potential failure modes identified throughout the safety analysis process.

3.1.8 Step 6: Update the iStar4Safety Models

Finally, in Step 6—the final step of each iteration—the analyst updates the iStar4Safety models to incorporate all the findings obtained throughout the previous steps of the RESafety process.

At this stage, elements such as safety constraints and responsibilities, previously modeled in Step 2 as *Safety Goals*, are revisited. Each responsibility should be linked to the higher-level safety constraint it contributes to fulfilling, thereby reflecting a refinement and operationalization of that constraint within the system model.

Loss scenarios are treated as *hazard causes*—situations or conditions that can lead to the occurrence of UCAs or HCs. Consequently, each loss scenario must be associated with the specific UCA or HC whose occurrence it contributes to, as established during its definition.

Safety requirements, in turn, are represented in the SR model as *Safety Tasks*. Each task must be explicitly linked to the loss scenarios it is designed to mitigate, ensuring traceability between identified hazards and their corresponding mitigation strategies within the model.

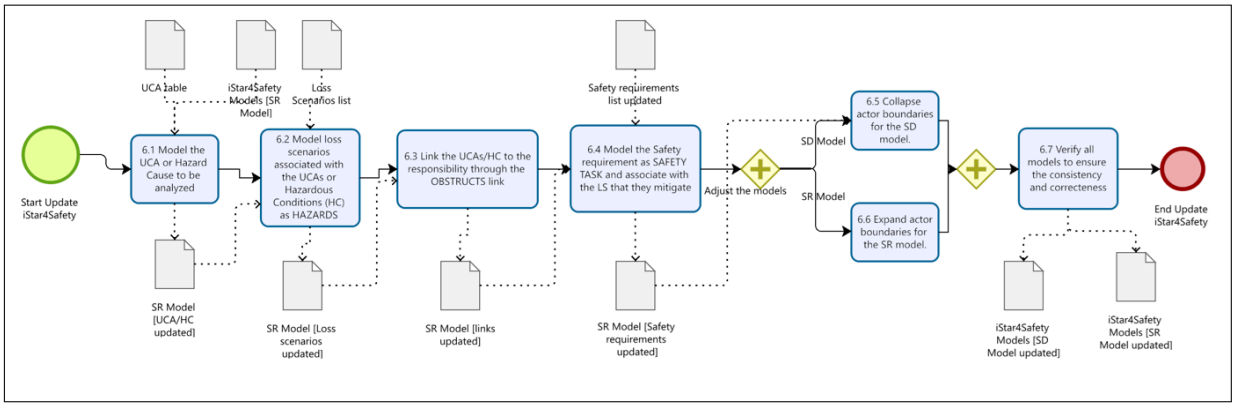
To facilitate traceability, all elements should have clearly defined and consistently applied identifiers. Since iStar4Safety models—particularly the SR model—can become quite large, it is recommended that each element be labeled with the identifier of the corresponding UCA, loss scenario, or safety requirement. Detailed descriptions of these elements should be maintained in the analysis documentation. When using the piStar4Safety tool, these descriptions can be stored in the description field provided for each element in the tool.

The SR model will incorporate the safety logic derived from the RESafety analysis. The SD model may also require adjustments, depending on how the SR model is refined or extended throughout the process.

However, it is important to emphasize that the SD model will not include safety-specific elements. This decision follows the iStar4Safety methodological guideline, which stipulates that for an element to be considered a safety element, it must be accompanied by the full safety reasoning structure: a *Safety Goal*, the *Hazard(s)* that obstruct it, and the corresponding *Mitigation Tasks*. Without this complete logic, the element is not classified as part of the safety analysis.

Figure 3.14 illustrates the activities involved in Step 7 of the RESafety process.

Figure 3.14: Step 7 - Update iStar4Safety Models - BPMN Representation of Step 7 Workflow



Source: Author (2025).

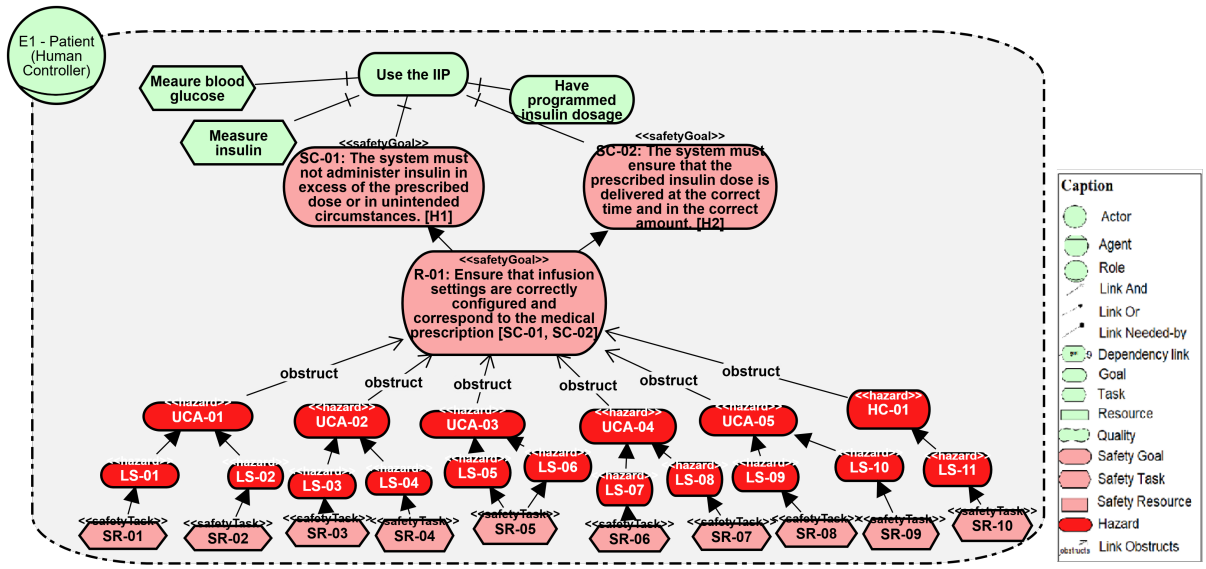
This step is initiated once UCAs and HCs have been analyzed and corresponding Loss Scenarios and Safety Requirements have been identified. The step comprises seven substeps, as detailed below:

- **6.1 Model the UCA or Hazardous Conditions to be analyzed:** The UCA or HC identified in previous steps is included in the SR model of iStar4Safety as an intentional element, safety task, that will be linked to a responsibility.
- **6.2 Model the Loss Scenarios associated with the UCAs or HCs as HAZARDS elements:** Each LS is added as a `Hazard` element within the SR model, maintaining traceability with the originating UCA/HC.
- **6.3 Link the UCAs/HCs to the responsibility through the `OBSTRUCTS` link:** This step establishes the relationship between the unsafe behavior (UCA/HC) and the system responsibility it endangers, using the "Obstructs" link of the iStar4Safety.
- **6.4 Model the Safety Requirements as `SAFETY TASKs`:** Each safety requirement derived from the LS analysis is modeled as a `SAFETY TASK`, which mitigates one or more LSs through the OR relationship.
- **6.5 Collapse actor boundaries for the SD model:** For the *Strategic Dependency (SD)* view, actor boundaries are collapsed.
- **6.6 Expand actor boundaries for the SR model:** For the *Strategic Rationale (SR)* view, actor boundaries are expanded to explicitly show internal reasoning elements.
- **6.7 Verify all models to ensure consistency and correctness:** A final review ensures that the SD and SR models are aligned, well-formed, and consistent with the safety analysis outcomes. Artifacts are then updated.

This systematic approach seeks to ensure that all previously identified safety concerns—whether originating from Unsafe Control Actions (UCAs) or Hazardous Conditions (HCs)—are properly reflected and traceable within the iStar4Safety models. It reinforces one of the objective of the RESafety process: to support the modeling of safety requirements while enabling traceability throughout all steps of the analysis.

IIP System Example: Figure 3.15 presents the updated SR model of the insulin infusion pump, which serves as the illustrative example in this chapter. Due to space limitations, only the elements related to the control action “Program insulin dosage” have been modeled. We emphasize that the full diagram will be significantly larger once all control actions, their associated UCAs, loss scenarios, and safety requirements are defined and modeled.

Figure 3.15: Strategic Rationale (SR) model of the IIP System updated in Step 6 of the RESafety process, illustrating the safety logic associated with the control action "Program insulin dosage".



Source: Author (2025).

Here, the reader can observe the layers that compose the initial and complete safety modeling of RESafety: the safety constraints and responsibilities that must be fulfilled to prevent losses; the Unsafe Control Actions (UCAs) and Hazardous Conditions(HCs), which, if they occur, may hinder the fulfillment of those safety constraints and responsibilities, thereby potentially leading to losses; the layer of loss scenarios (LSs), which represent the causes that trigger UCAs and HCs; and finally, the layer of safety requirements (SRs), which completes the safety logic by providing the necessary measures to mitigate the occurrence of such hazards.

3.1.9 Additional Considerations about RESafety

RESafety is a reference process designed to support the early modeling of safety requirements in safety-critical systems. It can be customized by analysts according to the characteristics

of the target system and the available safety-related information. We illustrated this customization in Section 3.2.

The RESafety process is structured into six steps, allowing flexibility in the modeling of responsibilities, safety constraints, unsafe control actions (UCAs), and the derivation of safety requirements. The customization involves selecting relevant elements from the control structure and tailoring the analysis to the specific operational context.

The BPMN diagram describing the RESafety process is available at: `https://monikysribeiro.github.io/resafety.html?lang=en`.

The next section presents a detailed view of how RESafety can be tailored in a real-world context.

3.2 Alternative Uses of the RESafety Process

RESafety is a process designed to support the development of the *Safety Analysis Document*. This artifact includes elements produced throughout each step of the process, namely:

- **STEP 1** – General Concerns and Safety Concerns
- **STEP 2** – Initial iStar4Safety SD and SR models
- **STEP 3** – System Safety Control Structure
- **STEP 4** – Table of Unsafe Control Actions (UCAs) and Hazardous Conditions (HCs) and Table of Controller Constraints
- **STEP 5** – Table of Loss Scenarios and Safety Requirements
- **STEP 6** – Final iStar4Safety SD and SR models of the iteration

Each element can be used as input for the subsequent steps. Therefore, it is important to note that, in some cases, certain elements may already be available. When this occurs, they may be reused, provided they align with the process. It is up to the analyst to determine whether adaptations are needed to ensure compliance with the RESafety process.

In the following subsections, two possible scenarios are presented to illustrate alternative ways of using RESafety. These are provided as suggestions, as the analyst must ultimately assess the specific characteristics of the system and its context to determine how best to apply each step of the process in a feasible and effective manner.

3.2.0.1 iStar Model Already Available

When a system already has an iStar model developed, it can serve as an input for Step 1, particularly for identifying General Concerns. For Step 1.2, which involves defining Safety Concerns, the model should be analyzed to identify elements with safety-critical attributes.

Once Step 1 is completed and the iStar models are available, Step 2 should be carried out using iStar4Safety, as it provides dedicated constructs to model the system's initial safety requirements. From this point, the analysis can proceed through the subsequent steps, according to the desired number of iterations.

3.2.0.2 STPA Analysis Already Available

In cases where a previous STPA analysis exists, the analyst should start by addressing Step 1.1 (General Concerns) and then create the iStar4Safety SD and SR models required for Step 2. This allows the modeling of additional non-safety requirements if desired. It is also important to consider that STPA analyses, especially those following the approach proposed by Leveson & Thomas (2018), typically do not include the modeling of Hazardous Conditions (HCs), which must therefore be developed in Step 4.8.

3.3 Information Management in the RESafety Process

Effective management of the information produced throughout the RESafety process is essential, given the iterative and evolutionary nature of safety analysis. Each step of the process generates specific artifacts, which must be properly organized, versioned, and updated to ensure consistency and traceability. The following guidelines outline how the artifacts of each step can be created, stored, and maintained.

3.3.1 Main Artifact (All Steps): Safety Analysis Document

The main artifact of the RESafety process is the **Safety Analysis Document**, which consolidates all analyses performed, including textual descriptions, tables, and diagrams. Any text editor that supports images and tables can be used for its creation. It is recommended to use collaborative tools such as Google Docs, which facilitate sharing, version control, and team co-editing.

3.3.2 Step 1 – Define the Scope of the SCS

For documenting the system scope, a text editor that allows the inclusion of tables may be employed. Examples include Google Docs or Notion, which provide flexibility and collaborative features suitable for recording the results of this step.

3.3.3 Step 2 – Define iStar4Safety Models (SD and SR)

For modeling the iStar4Safety diagrams, it is recommended to use the **PiStar4Safety** tool, which is open source and specifically designed to support the iStar4Safety language. This tool enforces modeling constraints, preventing invalid relationships and ensuring adherence to

language rules. Alternatively, general-purpose tools such as Draw.io may be used, provided they support all iStar4Safety elements, including the *obstructs* link.

3.3.4 Step 3 – Define the Control Structure

To model the control structure, the Draw.io online tool is recommended, as it provides flexibility for constructing system-theoretic diagrams. The commercial tool **Astah System Safety** can also be used; it natively supports STPA modeling and provides features for defining and visualizing control structures.

3.3.5 Step 4 – Identify Unsafe Control Actions (UCAs)

For this step, a simple text or spreadsheet editor can be used, since the results are presented in tabular format, listing UCAs and, when applicable, HCs. Astah System Safety also offers partial automation for this task, generating UCAs directly from previously defined control actions. Controller constraints can also be created using text or spreadsheet editors, as they are typically organized in tabular form.

3.3.6 Step 5 – Derive Safety Requirements

As in Step 4, a text or spreadsheet editor can be used to document UCAs and HCs associated with their respective loss scenarios and safety requirements.

3.3.7 Step 6 – Update iStar4Safety Models

The same modeling tools used in Step 2 should be applied here, since the SD and SR diagrams are updated to incorporate safety requirements and refined dependencies.

Versioning and Configuration Control

Regarding version control, the use of distributed configuration management tools such as Git is recommended to maintain separate versions for each iteration of the analysis. Proper versioning ensures that the evolution of the RESafety artifacts is systematically recorded and traceable. Each iteration should be clearly labeled to indicate its corresponding version and stage in the process. Maintaining this structure helps preserve key analysis versions, facilitates collaborative work, and supports the iterative refinement that characterizes the RESafety process.

3.3.8 Final Remarks

This information management proposal was suggested by the author as a practical guideline for maintaining consistency and accessibility of all artifacts generated during the

RESafety process. It is important to note that the suggested tools can be adapted according to team preferences and institutional constraints. As future work, the development of an integrated tool is proposed, capable of encompassing all RESafety steps and automating or semi-automating the creation and maintenance of its artifacts.

3.4 Chapter Summary

This chapter presented the RESafety process, composed of six steps that integrate elements from STPA and iStar4Safety. The steps were explained through a BPMN-based model and illustrated with the Insulin Infusion Pump System, a widely adopted example in the literature.

In the next chapter, the evaluations conducted in this research are presented. The first consists of illustrating the process through the analysis of a real system—a medication delivery system that employs a robotic arm in a hospital context. The second comprises an expert-based evaluation involving safety engineering and requirements engineering specialists, carried out using the TAM (Technology Acceptance Model) framework.

4

Application of the RESafety Process in a Robotic Medicine Delivery Scenario

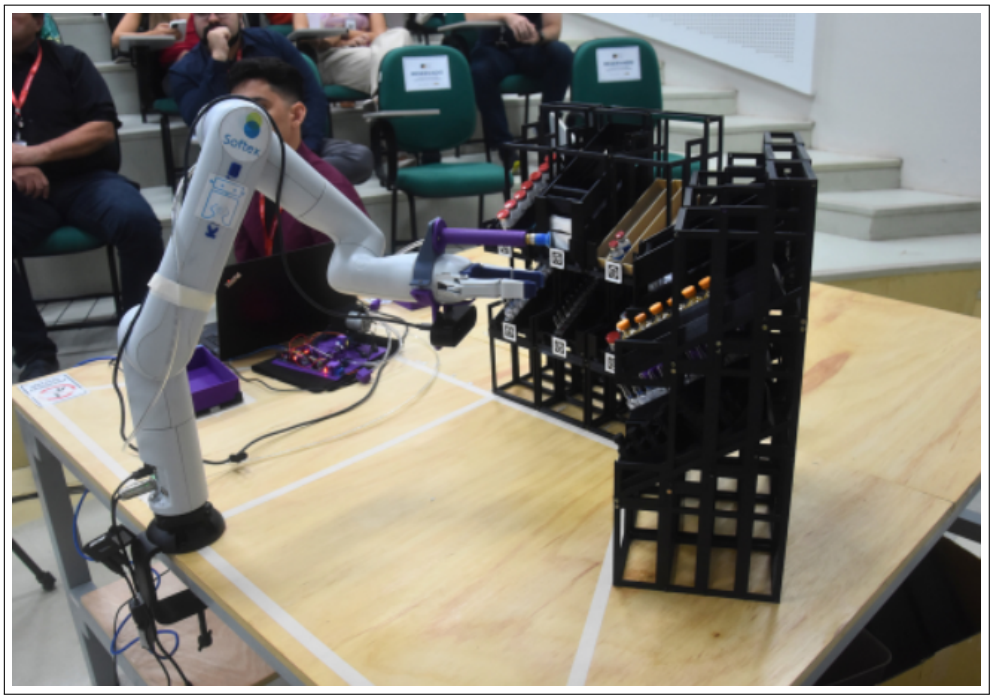
This section presents the application of the RESafety process in a real-world case study involving a robotic arm used for drug dispensing in a hospital pharmacy environment. The objective is to demonstrate how the proposed process can support the modeling of safety requirements in the context of safety-critical systems that involve human-robot interaction.

4.1 Study Site and Scenario Description

The Kinova Gen3Lite robotic arm is classified as a collaborative robot (cobot), as it operates without the need for physical barriers separating it from humans (Adriaensen *et al.*, 2021). The most accurate classification for its interaction with humans is "coexistence," in which both entities share the same physical space but do not enter each other's operational zones (Wang *et al.*, 2019; Adriaensen *et al.*, 2021). Although existing standards for collaborative robots do not define specific assessment methods, recent studies have employed STPA to analyze hazardous scenarios in human–robot collaborative systems. In this way, STPA was not directly applied as a requirements elicitation technique but rather as a hazard analysis approach that provides the foundation for defining safety-related requirements and trade-offs between safety and mission efficiency.(Adriaensen *et al.*, 2021).

The robotic arm in question remains stationary, mounted on a table that also holds the medication shelves. The operational range of the arm includes access to pre-packaged medications positioned on these shelves in front of the robot (see Figure 4.1).

Figure 4.1: Robotic Medicine Delivery System



Source: Hospital das Clínicas da UFPE (2023)

This study was conducted during the author's participation in a Software Residency Program developed by the Softex company at the Center for Informatics (CIn), Federal University of Pernambuco (UFPE). The project focused on advanced development and testing of a robotic system for drug delivery using the Kinova Gen3Lite robotic arm.

The implementation target for this system was the Hospital das Clínicas in Recife/PE, managed by the Brazilian company Ebserh[1].

The author of this dissertation joined the project as a resident for seven months, during which she was trained in the physical handling and programming of the robot using Python. The program also included coursework on robotics and artificial intelligence. The RESafety modeling presented here was initiated during this residency, providing access to the robot, developers, professors, residents, and hospital pharmacy staff.

The information regarding the current medication dispensing process was clarified through an interview with an pharmacy technician—whose name is anonymized for privacy reasons— pharmacy where the robotic arm would be deployed and tested. On average, around 250 patients are treated daily, generating approximately the same number of prescriptions, each valid for 24 hours (from midday to midday of the following day). Prescriptions may be updated or modified throughout the day as needed. The pharmacy receives medication requests and sends them to the nursing team, which returns any drugs suspended by the physician.

[1]More information is available at: `https://tinyurl.com/u257ntre`

4.2 Operation Flow Before and During the Deployment

To understand the impact of robotic automation on the medicine dispensing process, two scenarios were analyzed: the current operational workflow (*System-as-is*) and the projected workflow after the robot's integration (*System-to-be*).

4.3 System-as-is

Before the deployment of the robotic system[2], the pharmacy workflow relied entirely on manual operations performed by human staff. An interview with a pharmacy technician provided valuable insights into the current procedures and expectations regarding the potential integration of automation technologies. Although the robotic arm was tested during a pilot phase, it was removed following the conclusion of the Softex project activities.

The current medication dispensing workflow begins with the physician issuing a prescription in the AGHU system based on the patient's clinical condition. This prescription is routed to two sectors: (i) the Pharmacy, where it becomes available on the AGHU "dispensing screen," allowing the team to prepare and deliver medications, and (ii) Nursing, which receives the medications physically from the pharmacy and is responsible for administering them to the patient, recording the administration on a printed copy. The main objective of deploying the robotic arm is to support and optimize the dispensing process carried out by the pharmacy team.

Within the pharmacy, prescriptions generated in the AGHU system are validated by physicians and subsequently reviewed by a pharmacist, who performs a detailed triage to confirm the appropriate dosage, administration schedule, and any specific restrictions. After this validation, the prescription moves to the dispensing stage, where a pharmacy technician physically selects the medications, packages them, and places them in individually labeled bags for each patient.

Previously, the process was more centralized: the pharmacist was responsible for triage, dispensing, and printing the ticket, leaving the technician solely in charge of collecting the ticket and physically separating the medications. The current routine has been adjusted to distribute tasks more efficiently: the pharmacist performs triage in the system, the prescription is then forwarded to a second "dispensing screen," where a technician validates the quantities, prints the ticket, and either the same technician or another one completes the physical separation of the medications.

We present in the next section the proposed system configuration, now considering the integration of the robotic arm to perform the dispensing task—that is, retrieving medications from the shelves and placing them into the appropriate patient bag according to the corresponding ticket.

[2]The Gen3Lite robot was not permanently installed due to budgetary and strategic planning constraints.

4.4 System-to-be: Integrating Gen3Lite into the Medicine Dispensing Workflow

The deployment of the Gen3Lite robotic arm aims to support the dispensing process by automating the retrieval of medications from the shelves arranged around the robot on a dedicated table. The shelves are positioned within the robot's reach and designed with three levels to accommodate medications of different sizes. As demonstrated in the presentation video[3], the shelving units rely on gravity to facilitate the movement of items toward the picking area, enabling easier access for the robotic arm.

During the initial project phase observed, dispensing was performed for one patient bag at a time, requiring manual collection after each ticket was processed. However, the intended goal, as illustrated in the video, is to allow up to 18 patient bags to be filled before retrieval is necessary.

The shelves are stocked with medications identified by specific markers, which are read using computer vision. The robotic arm can handle different types of packaging: medicine bags are collected via a suction device, while ampoules are grasped using the robot's gripper. Error handling is currently performed by a human operator, who receives alerts from the robot's interface whenever a predefined issue is detected, along with suggested corrective actions. At the time of observation, no formal safety analyses had been conducted for this system.

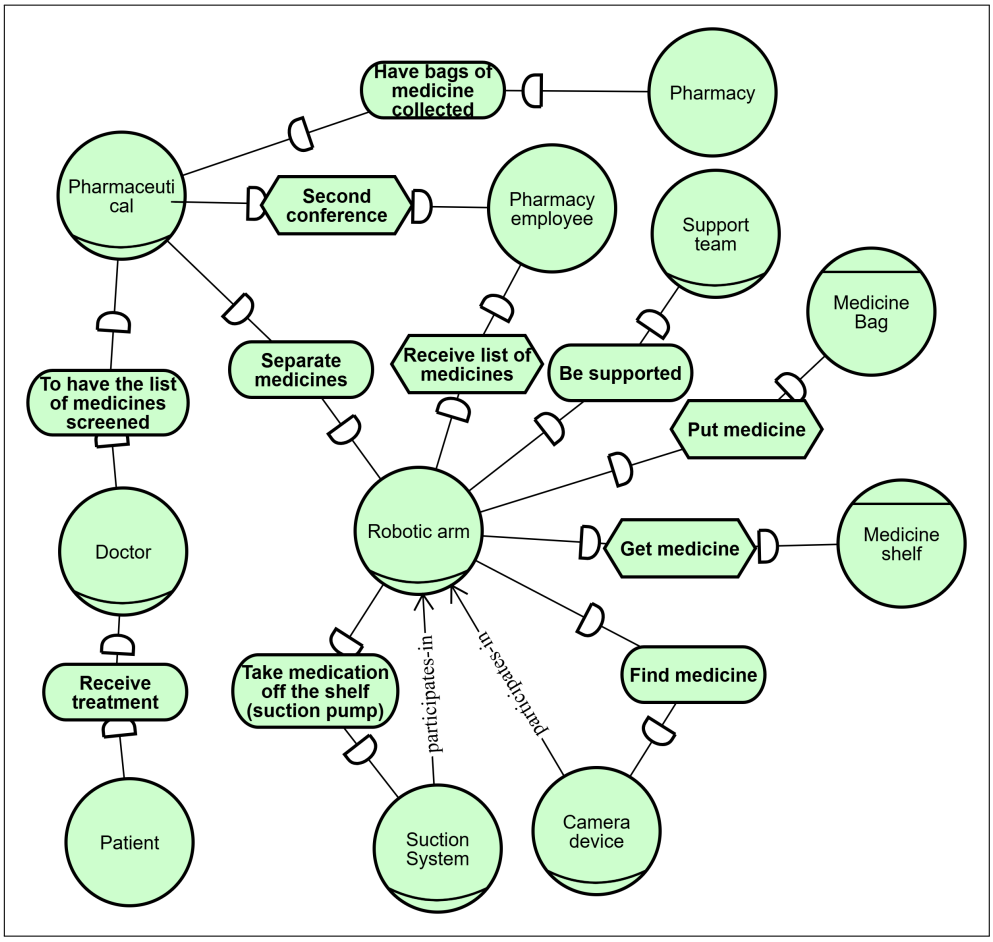
For safety analysis purposes, the scope was limited to the robotic arm subsystem. A high-level control structure was first introduced to illustrate the system architecture. In addition, an iStar4Safety SD model was developed to represent the overall operation and stakeholder responsibilities.

It is important to note that this control structure focuses only on elements relevant to the robotic dispensing scenario. As emphasized by Leveson (2011), additional processes could be integrated at higher control levels, such as certification and regulatory policies. For this case study, an abstracted structure was adopted, focusing on hazards directly related to the robot's operation while treating higher-level influences as environmental inputs. This top-down approach is consistent with Leveson's recommendation to initiate safety analysis by identifying system-level hazards and then tracing their causes through the control structure.

A preliminary SD model was created to support the initial understanding of the context in which the robotic system is inserted, and it can be seen in Figure 4.2.

[3]Available at: `https://www.youtube.com/watch?v=h1guOAtFBUg`

Figure 4.2: Preliminary SD model



Source: Author (2025).

Figure 4.2 illustrates the context of the pharmacy system and its process at the time of the robotic arm installation.

The model includes the *Patient* actor, who depends on the *Doctor* to achieve the goal **"Receive treatment"**. The *Doctor*, in turn, relies on the *Pharmaceutical* to accomplish the goal **"Have the list of medicines screened"**. The *Pharmaceutical* actor, responsible for performing the initial screening of the medicine ticket generated by the doctor, depends on the *Pharmacy Employee* to carry out a **"Second verification"** of the list of medicines previously screened. Additionally, the *Pharmaceutical* relies on the *Robotic Arm* to achieve the goal **"Separate medicines"**.

The *Robotic Arm*, in turn, depends on the *Pharmacy Employee* to **"Receive list of medicines"** to be processed. It also depends on the *Support Team* to **"Be supported"**, should technical assistance be required.

The *Robotic Arm* depends on the *Medicine Bag* to **"Put medicine"** into it — this compartment stores the dispensed medications for later retrieval by the *Pharmaceutical*, who in turn depends on the *Pharmacy* to **"Have bag of medicine collected"**.

The *Robotic Arm* also depends on the *Medicine Shelf* to **"Get medicine"**, as the robot

must move to the correct shelf compartment to retrieve the desired item. Two components are directly integrated with the robotic arm: the *Suction System*, which handles the suction of medications stored in flexible packaging — modeled as the dependency **"Take medication off the shelf (suction pump)"** — and the *Camera Device*, which the robot depends on to **"Find medicine"**. This last task involves reading positional markers on the shelf or medicine package, such as tags or QR codes.

We would like to highlight that this initial modeling aims to provide the reader with an understanding of the robotic arm's operation within the general hospital context, based on observations and gathered reports. However, in the system illustration presented in the next section, the focus is on modeling a high-level view of the robotic system, which includes only three actors: *Pharmacy Employee*, *Robotic Arm*, and *Support Team*. This choice reflects the intention to maintain a simplified representation, with the understanding that subsequent iterations of the RESafety process will allow for further specification of the system while preserving the initial contextual details.

4.5 Application of the RESafety Process

The RESafety process was applied in the following sequence of steps:

- **Step 1 – Identify General and Safety Concerns:** Identification of the main objectives and potential safety concerns of the Robotic Medicine Delivery System.
- **Step 2 – Initial iStar4Safety Models (SD and SR):** Modeling of stakeholder goals, responsibilities, and safety concerns using the iStar4Safety language.
- **Step 3 – Control Structure Modeling:** Development of a STPA-based control structure.
- **Step 4 – Unsafe Control Actions (UCAs):** Identification and documentation of UCAs based on the STPA methodology and Hazardous Conditions. Derivation of safety constraints aimed at mitigating or eliminating the identified UCAs and HCs.
- **Step 5 – Loss Scenarios and Safety Requirements:** Elaboration of potential loss scenarios and formulation of corresponding safety requirements.
- **Step 6 – Final iStar4Safety Models:** Update of the SD and SR models to incorporate the UCA and HCs, along with their corresponding Loss Scenarios and the Safety Requirements that mitigate them.

Each step of the process is detailed in the following sections, along with the resulting artifacts and reflections on the modeling experience.

4.6 Stepwise Execution of the RESafety Process

Each step of the RESafety process is detailed in the following subsections, presenting the modeling decisions, generated artifacts, and key reflections.

4.6.1 Step 1 – Define the scope of the safety critical system (SCS)

This section presents the General Concerns and Safety Concerns identified in the analysis of the Robotic Medicine Delivery System.

- **1.1 - General Concerns**
  - **1.1.1 - Analysis Objectives:** This analysis aims to model a drug delivery process using a robotic arm in the context of a hospital pharmacy.
  - **1.1.2 - System Definition:** The system consists of a robotic arm applied to the medication dispensing process in the hospital pharmacy. The arm is designed to retrieve medicines using either a suction device (for medicine bags) or a gripper (for ampoules) from designated shelves and place them into patient-specific bags, fulfilling the medication tickets generated for hospitalized patients.
  - **1.1.3 - Resources Needed for Analysis:** The following resources were consulted or are associated with the robotic medication delivery system project:
    - Pharmacy visit report *(Limited access)*
    - Notion app page with general project information and activity tracking *(Limited access)*
    - Project page on GitHub *(Limited access)*
    - Video – AGHUX Training (Version 10) - Pharmacy Module
    - News – HC Innovation Showcase
    - HC Project Presentation
    - Case Study – STPA Analysis of an Industrial Cooperative Robot and an Autonomous Mobile Robot
    - Case Study – STPA Safety Analysis of a Collaborative Robot Application
    - Comparative Study – STPA Hierarchical Structures in Risk Analysis – Complex Multi-Robot Mobile Systems
  - **1.1.4 - System Boundary:** The main process begins once the patient ticket is issued in the dispensing system by the pharmacist and concludes when the medication is placed into the designated dispenser bag.

- **1.1.5 - System Components:**
  - Robotic arm (Gen3 Lite)
  - Pharmacy Employee
  - Support team

- **1.2 - Safety Concerns**

  - **1.2.1 - Losses:** A loss refers to any negative outcome that affects something stakeholders consider valuable. Such outcomes may include fatalities or injuries, damage to property, environmental harm, mission failure, reputational damage, leakage of sensitive information, or any other consequence deemed unacceptable by stakeholders (Leveson & Thomas, 2018). It is important to highlight that, according to Leveson & Thomas (2018), in industry practice the terms losses, accidents, mishaps, and even adverse events are often used interchangeably, which can lead to confusion. In this work, the term loss will be used in order to allow a more generalizable concept. In this approach, losses are labeled using the prefix **"Lx"**. References to individual components or specific causes, such as "human error", must be avoided when defining losses (Leveson & Thomas, 2018).
    - **L1** - Risk of life
    - **L2** - Risk of injury
    - **L3** - Mission failure
    - **L4** - Equipment damage
    - **L5** - Financial loss
  - **1.2.2 - System-Level Hazards:** Hazards are system states or set of conditions that in the case of a worst-case environmental condition will lead to losses (Leveson & Thomas, 2018). When identifying hazards, the analyst must specify the associated losses, ensuring traceability between hazards and losses. Furthermore, suppose the analyst already has UCAs, identified from a prior STPA analysis or earlier iterations; verifying whether these UCAs are linked with defined hazards is imperative. Otherwise, analysts must specify a new hazard related to the UCA. This is because UCAs are also the causes of hazards. Avoid referencing individual components in the hazard definition.
    - ***H1*** - Robotic arm delivers the wrong treatment [L1, L2, L3]
      - *H1a* - Right patient, wrong medicine
      - *H1b* - Right patient, wrong quantity
      - *H1c* - Wrong patient

- ***H2*** - Robotic arm violates minimum separation distance from humans or objects [L2, L3, L4] (e.g., collision risk)
- ***H3*** - Robotic arm breaks or tears the medication [L3, L5]
- ***H4*** - Robotic arm loses medicine [L3]
- ***H5*** - Robotic arm operated beyond limits [L2, L3, L4]
- ***H6*** - Loss of robotic arm control [L2, L3, L4]
- ***H7*** - The integrity of the robotic arm structure is compromised [L4, L5]
- ***H8*** - Robotic arm is prevented from delivering medicines [L3]

- **1.2.3 - Safety Constraints:** In this step of the process, the safety constraints for the identified hazards must be defined. It is important to note that, at this point, the focus is not on defining specific solutions or implementations for hazard mitigation, but rather on establishing high-level constraints that must not be violated by the system. These constraints can be formulated as simple negations of the hazards and should be traceable to one or more hazards (Leveson & Thomas, 2018).
  - **SC-01** - The Robotic arm must deliver the correct medication to the right patient [H1]
  - **SC-02** - The Robotic arm must not exceed the minimum separation distance from humans and objects [H2, H3, H4, H6, H7]
  - **SC-03** - The Robotic arm must not break or tear the medication [H3]
  - **SC-04** - The Robotic arm must not lose the medication [H4]
  - **SC-05** - The Robotic arm needs to operate under the conditions defined for its best functioning [H5, H6, H7]
  - **SC-06** - The robotic arm needs to be under control throughout its use [H6, H7]
  - **SC-07** - The structural integrity of the robotic arm must be maintained [H7]
  - **SC-08** - Robotic arm needs to deliver medicine when needed [H8]
- **1.2.4 - Responsabilities:** At this point, the analyst must define the responsibilities of each system actor or component to ensure that the established safety constraints are effectively pursued. These responsibilities represent refinements of the safety constraints and specify what each entity must do

to help enforce them. It is recommended that each component be systematically reviewed against every safety constraint, identifying its specific responsibilities—if any—in ensuring that the constraint is enforced. See Table 4.1 for the system under consideration.

Table 4.1: Responsibilities of each entity involved in the drug dispensing scenario

Entity	Responsibility
Pharmacy Employee (PE)	**R-01:** Ensure that the patient's ticket is correct [SC-01] **R-02:** Ensure that the patient's medicine bag is correctly positioned before dispensing [SC-01] **R-03:** Do not enter the robot's operational area while it is moving [SC-02] **R-04:** Restart ticket delivery if the medicine bag was not completed after dispensing [SC-03, SC-04] **R-05:** Observe and promptly report any abnormal behavior of the robotic arm to the Support Team [SC-03, SC-04, SC-05, SC-06, SC-08] **R-06:** Periodically check the structural integrity of the robotic arm and report anomalies [SC-07]
Support Team	**R-07:** Diagnose robotic arm failures and authorize operation only after restoring safe functionality [SC-01, SC-05] **R-08:** Be available to respond to robotic arm operation emergencies [SC-05] **R-09:** Interrupt robotic arm operation if unsafe conditions are detected [SC-05] **R-10:** Perform preventive maintenance inspections on the robotic arm [SC-07] **R-11:** Approve robotic arm operation only after verifying structural integrity [SC-07]

Continued on next page

Table 4.1 – continued from previous page

Entity	Responsibility
Robotic Arm	**R-12:** Ensure delivered medication matches ticket [SC-01] **R-13:** Avoid abrupt or non-standard movements [SC-02] **R-14:** Grasp/suction medicine according to operational parameters [SC-03] **R-15:** Maintain grip on medication until dispensing into the designated bag [SC-03] **R-16:** Monitor medication possession and signal anomaly if lost [SC-03, SC-04] **R-17:** Release medication into bag at a safe distance [SC-03] **R-18:** Move according to defined operational instructions [SC-05, SC-06] **R-19:** Generate operational logs to support anomaly detection and system behavior verification [SC-05, SC-06] **R-20:** Emit alerts upon detecting malfunction [SC-05, SC-06] **R-21:** Avoid movements that damage structural integrity [SC-07]

Source: Author (2025).

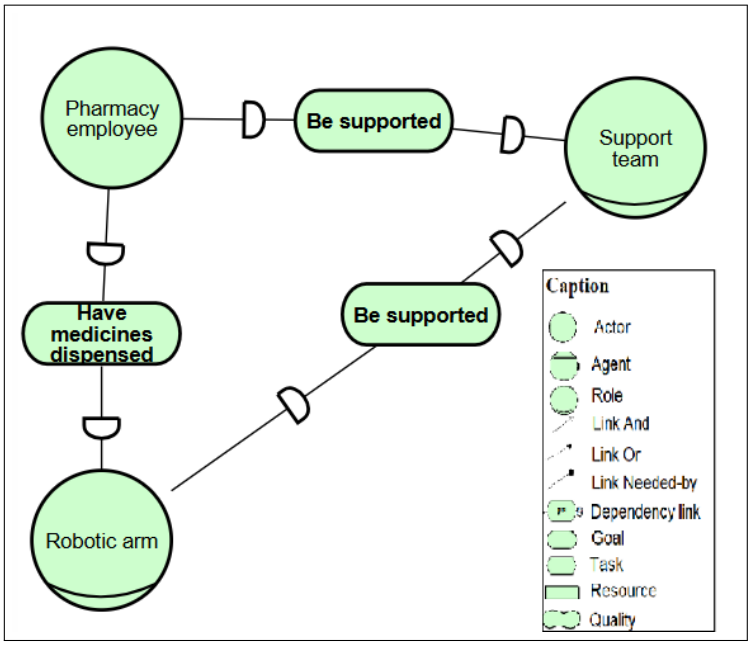
- **1.2.5 - Define other relevants artefacts:** Finally, the analyst may define relevant artifacts that can be used or are already available and relate to the system's safety. **Not applicable at this time.**

4.6.2 Step 2 – Define the iStar4Safety Models

1. **2.1 - SD Model:** See Figure 4.3. The model presents an initial scenario modeled for the medication delivery system using a robotic arm. As the reader can observe, there are three actors: the **Pharmacy employee**, the **Support team**, and the **Robotic arm**[4].

[4]The robotic arm was represented as a *role* because the specific robotic arm to be used depends on design choices. The real-world example employed the Kinova model, but this is not a mandatory requirement. The **Pharmacy employee** depends on the **Robotic arm** to achieve the goal **Have medicines dispensed**. The **Pharmacy employee** also depends on the **Support team** to **Be supported** when needed. Finally, the **Robotic arm** also depends on the **Support team** to **Be supported** whenever such support is required.

Figure 4.3: Initial SD Model



Source: Author (2025).

2.2 - SR Model: After the refinement phase, the SR model shown in Figure 4.4 was developed and is presented here. As can be seen, each actor has its overall goal defined when interacting with the system.

Figure 4.5 presents, for better visualization, the excerpt representing the *Pharmacy Employee* actor. This actor has the overall goal of **"Have tickets dispensed"**. Associated with this goal are all the Safety Constraints addressed by this actor's responsibilities, as defined in Step 1.2.3 — namely, SC-01 through SC-08. Each SC is linked to the responsibility or responsibilities through which the actor attempts to ensure that the associated constraint is satisfied. For the *Pharmacy Employee* actor, responsibilities R-01 through R-06 were modeled, as defined in Step 1.2.4, and are respectively associated with the corresponding SCs. Additionally, some non-safety elements were modeled in this step. The goal **"Be supported"** is fulfilled through a dependency on the *Support Team* actor, and the task **"Start ticket dispensing"** — at this point — is fulfilled by the *Robotic Arm* actor, whose main goal is **"Dispense medicines"**.

Continuing the SR model description by actor, Figure 4.6 presents the excerpt representing the *Support Team* actor. This actor has the overall goal of **"Provide technical support for system operation"**. Associated with this goal are the Safety Constraints addressed by this actor's responsibilities, as defined in Step 1.2.3 — specifically, SC-01, SC-05, and SC-07. Each SC is linked to the responsibility or responsibilities through which the actor aims to ensure that the corresponding constraint is satisfied. For the *Support Team* actor, responsibilities R-07 through R-10 were modeled, as defined in Step 1.2.4, and are respectively associated with the related SCs. In this case, there was no need to model additional non-safety elements at this stage of the iteration.

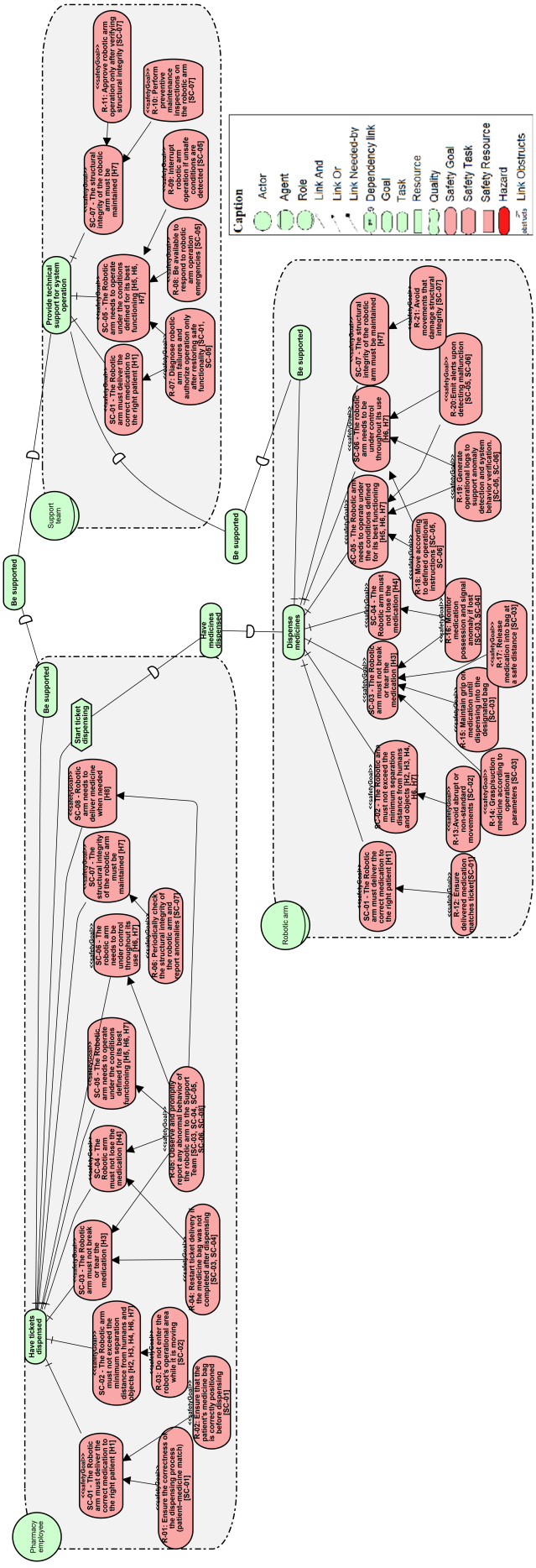


Figure 4.4: SR Model from Step 2 – General Overview

Source: Author (2025).

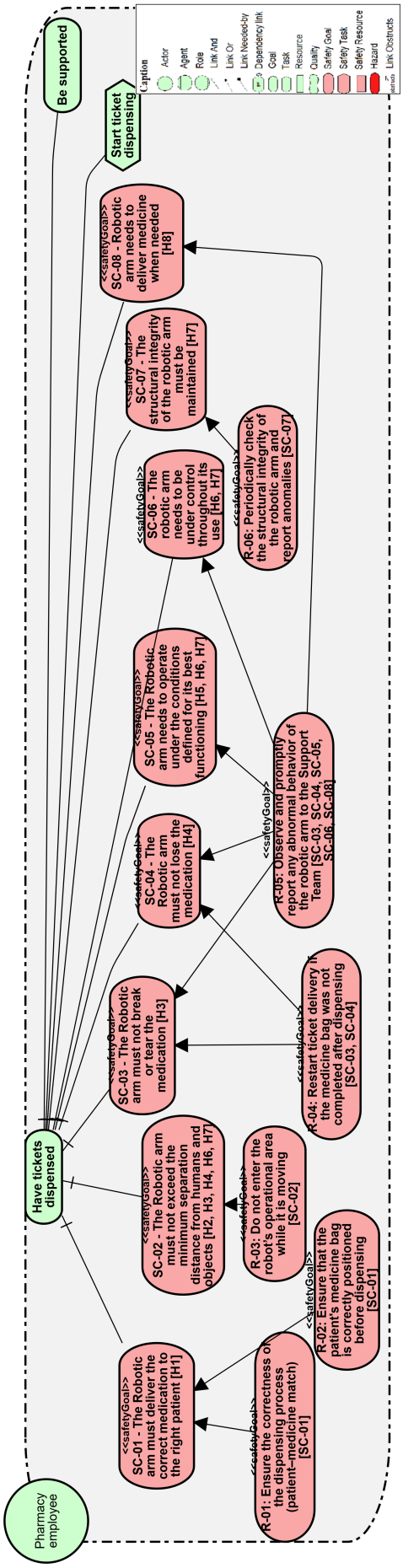


Figure 4.5: SR Model Excerpt from Step 2 – Pharmacy Emmployee Actor

Source: Author (2025).

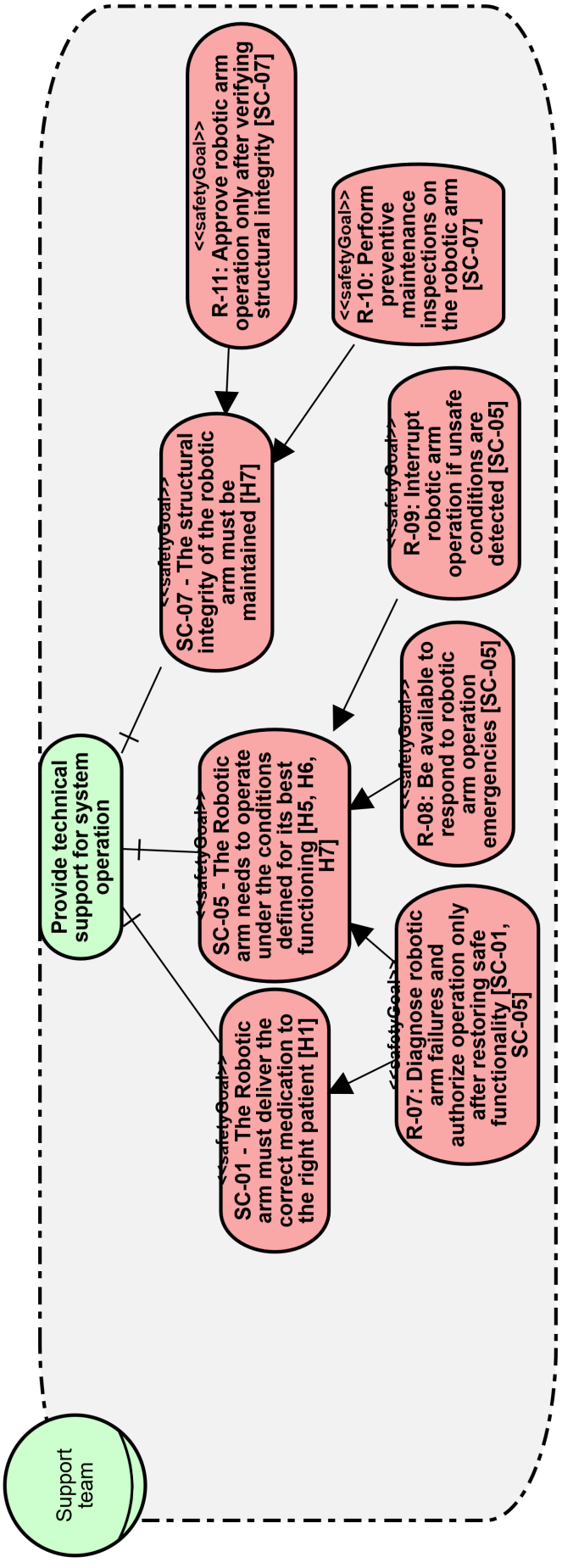


Figure 4.6: SR Model Excerpt from Step 2 – Support Team Actor

Source: Author (2025).

Finally, the detailed excerpt of the *Robotic Arm* actor is presented, as shown in Figure 4.7. Following the same structure used for the previous actors, this actor has the overall goal of **"Dispense medicines"**. Associated with this goal are the Safety Constraints addressed by this actor's responsibilities, as defined in Step 1.2.3 — namely, SC-01 through SC-07. Each SC is linked to the responsibility or responsibilities through which the actor attempts to ensure that the associated constraint is satisfied. For the *Robotic Arm* actor, responsibilities R-12 through R-21 were modeled, as defined in Step 1.2.4, and are respectively associated with the corresponding SCs. Additionally, one non-safety element was modeled at this step: the goal **"Be supported"**, which is fulfilled through a dependency on the *Support Team* actor.

4.6.3 Step 3 – Define the Control Structure

Figure 4.8 illustrates the control structure defined for the robotic arm-based medication dispensing system.

As can be observed, the three actors defined in **Step 2 – Define the iStar4Safety Models**—Pharmacy Employee, Support Team, and Robotic Arm—have been modeled. The key addition here is that the *Robotic Arm* actor has been decomposed into three elements: the *Robotic Arm Controller*, the physical *Robotic Arm* itself, and the controlled processes, namely the *Medicine Bag* and the *Medicine Shelf*.

This modeling choice reflects the need to distinguish the robot's controller from the physical device, and to inidcate that the other elements represent passive processes — components that are only acted upon, without providing feedback. The reader may note that the *Medicine Bag* and *Medicine Shelf* do not return any feedback, as they are merely physical components in the system: the *Medicine Bag* is an inanimate box located near the robot, and the *Medicine Shelf* is a structure where the medications are stored.

It is important to highlight that all these elements are collectively considered part of the *Robotic Arm* actor. The physical *Robotic Arm* includes the subcomponents: *Camera*, *Gripper*, and *Suction Device*. These are used by the robot to perform the task of retrieving medicines: the *Gripper* is used to grasp vial-type medications, the *Suction Device* is used for flexible-package medications, and the *Camera* identifies the tag of the medicine and/or the shelf.

Therefore, as shown, for the main processes — *Pharmacy Employee*, *Support Team*, and *Robotic Arm* — both the control algorithm and process model were defined. For the *Robotic Arm System*, these elements were modeled specifically for its controller, the *Robotic Arm Controller*.

It is worth noting that, since this is the first iteration, these elements may evolve as the project progresses. Furthermore, additional sensors, actuators, and subcomponents may be added as needed. It is also worth noting that responsibilities can be associated with UCA and other elements of the control structure, as the goal is to ensure their fulfillment.

The following additional observations are necessary: For analytical purposes, *"Give support"* was used here instead of *"Be supported"*, as modeled in the SR model, since it better

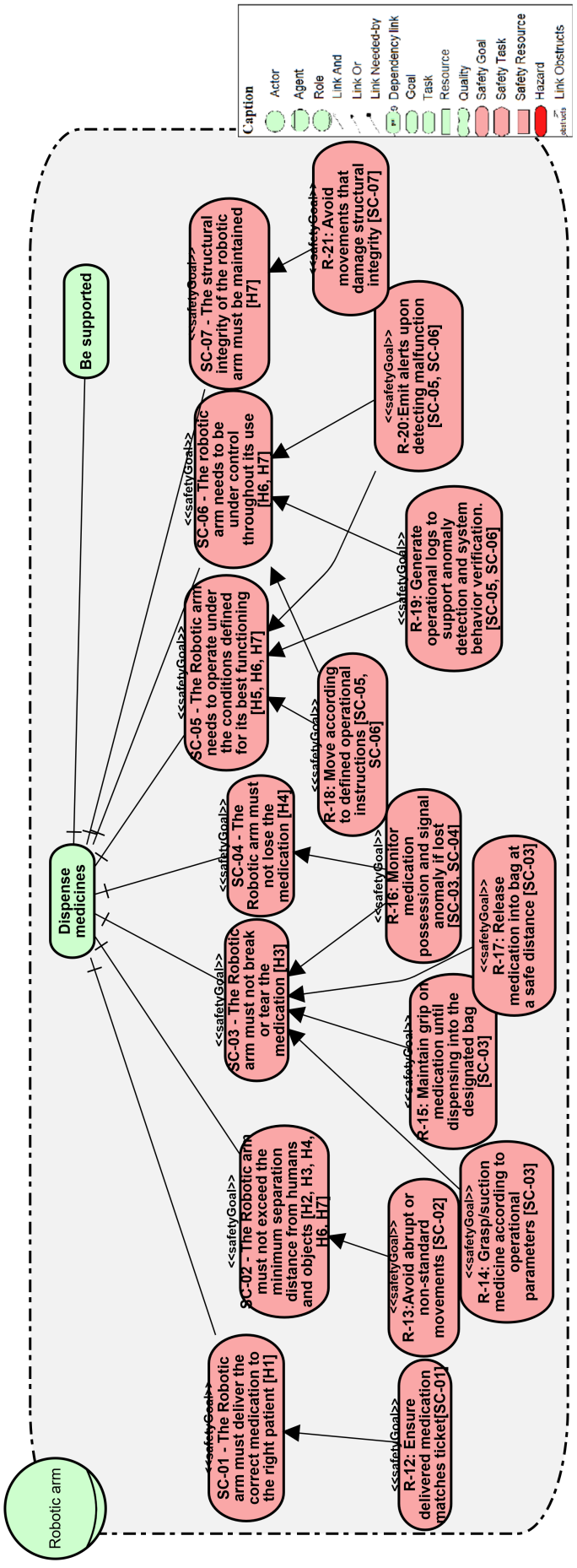
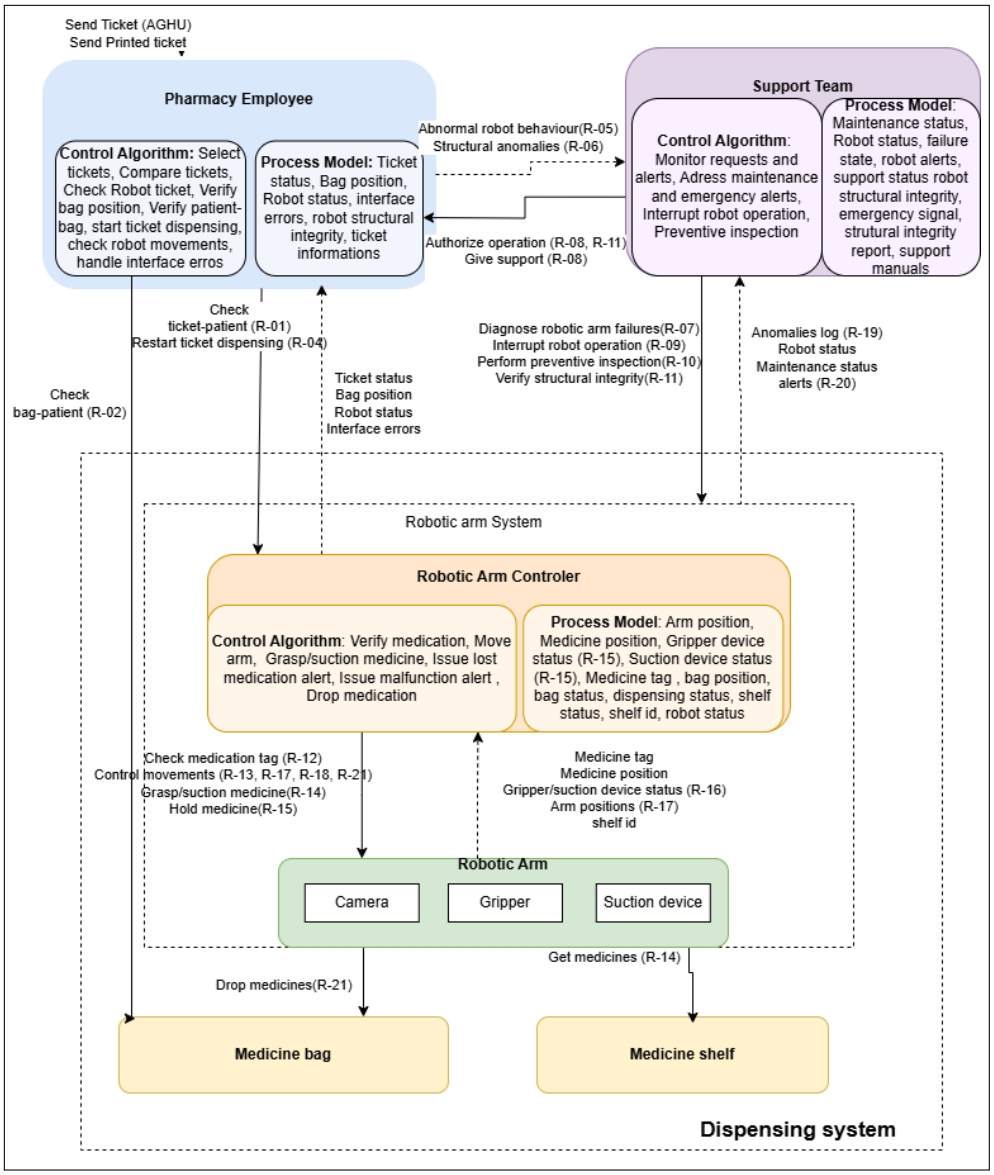


Figure 4.7: SR Model Excerpt from Step 2 - Robotic Arm Actor

Source: Author (2025).

aligns with the structure of this representation. Responsibility R-03 is addressed through feedback on the robot's status — if the robot is in motion, R-03 must be fulfilled.

Figure 4.8: Control structure of the robotic arm-based medication dispensing system



Source: Author (2025).

4.6.4 Step 4 – Identify Unsafe Control Actions (UCAs)

Next, the control actions were analyzed in terms of their potential to be unsafe. Accordingly, Table 4.2 presents theUCA identified for this system during the first iteration, along with the hazards to which they are associated.

As previously explained, the association with a hazard occurs because the control action fulfills a specific responsibility, which in turn aims to satisfy a Safety Constraint designed to address that hazard.

Each control action from the control structure defined in Step 3 was analyzed regarding its potential to be unsafe. In the case of the robotic arm medicine delivery system, all control actions were found to have the potential to be unsafe and were therefore examined against the four possible categories of UCAs: (i) if not provided, it could lead to a hazard; (ii) if provided, it could lead to a hazard; (iii) if provided at the wrong time, meaning too early, too late, or out of order; and (iv) if provided for an incorrect duration, meaning applied for less time than necessary or for too long. The results are presented in Table 4.2.

Table 4.2: Unsafe Control Actions (UCAs) – Step 4

Control Action	**From/To**	**Not Providing**	**Providing**	**Too Early, Too Late, Out of Order**	**Stopped Too Soon, Applied Too Long**
Check ticket-patient (R-1)	Pharmacy employee/Robotic arm controller	UCA-01: Pharmacy Employee does not provide "Check ticket-patient" before starting dispensing [H1]	UCA-02: Pharmacy Employee provides "Check ticket-patient" with incorrect or outdated data [H1]	UCA-03: Pharmacy Employee provides "Check ticket-patient" too late in the dispensing process [H1] UCA-04: Pharmacy Employee provides "Check ticket-patient" after the medication has already been dispensed [H1]	Not Applicable
Restart ticket dispensing (R-4)	Pharmacy employee/Robotic arm	UCA-05: Pharmacy Employee does not provide "Restart ticket dispensing" after failure to complete dispensing [H8]	UCA-06: Pharmacy Employee provides "Restart ticket dispensing" unnecessarily [H1]	UCA-07: Pharmacy Employee provides "Restart ticket dispensing" and the robotic arm is not prepared yet [H8]	Not Applicable
Check bag-patient (R-2)	Pharmacy employee/medicine bag	UCA-08: Pharmacy Employee does not provide "Check bag-patient" before dispensing [H1c]	UCA-09: Pharmacy Employee provides "Check bag-patient" with wrong assignment confirmation [H1c]	Not Applicable	Not Applicable
Diagnose robotic arm failures (R-7)	Support team/Robotic arm system	UCA-10: Support Team does not provide "Diagnose robotic arm failures" after detecting abnormal behavior [H1-H8]	UCA-11: Support Team provides "Diagnose robotic arm failures" with wrong conclusions [H1-H8]	UCA-12: Support Team provides "Diagnose robotic arm failures" too late [H1-H8]	UCA-13: Support Team provides "Diagnose robotic arm failures" but stops before completing the analysis [H1-H8]
Interrupt robot operation (R-9)	Support team/Robotic arm system	UCA-14: Support Team does not provide "Interrupt robot operation" when the robot is malfunctioning [H1-H8]	UCA-15: Support Team provides "Interrupt robot operation" when it is not necessary [H8]	UCA-16: Support Team provides "Interrupt robot operation" too late [H1-H8]	Not Applicable
Perform preventive inspection (R-10)	Support team/Robotic arm system	UCA-17: Support Team does not provide "Perform preventive inspection" as scheduled [H5, H7]	Not Applicable	Not Applicable	Not Applicable

Control Action	From/To	Not Providing	Providing	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Verify structural integrity (R-11)	Support team/Robotic arm system	UCA-18: Support Team does not provide "Verify structural integrity" of robotic arm when it is necessary[H5, H7]	UCA-19: Support Team provides "Verify structural integrity" using incomplete or invalid data [H7, H8]	Not Applicable	Not Applicable
Check medication tag (R-12)	Robotic arm controller/Robotic arm	UCA-20: Robotic Arm Controller does not provide "Check medication tag" before approaching the medication for grasping or suction [H1, H8]	UCA-21: Robotic Arm Controller performs "Check medication tag", but the tag is associated with the wrong item [H1]	UCA-22: Robotic Arm Controller performs "Check medication tag" prematurely, resulting in the detection of the wrong medication [H1] UCA-23: Robotic Arm Controller performs "Check medication tag" too late, causing the medication not to be detected [H8]	UCA-24: Robotic Arm Controller provides "Check medication tag" but ends before full read [H8]
Control movements (R-13, R-17, R-18, R-21)	Robotic arm controller/Robotic arm	UCA-25: Robotic Arm Controller does not provide "Control movements" while the robotic arm is in motion [H2, H5, H6, H7, H8]	UCA-26: Robotic Arm Controller provides "Control movements" in incorrect sequence [H4, H6, H8]	Not Applicable	Not Applicable
Grasp/ suction medicine (R-14)	Robotic arm controller/Robotic arm	UCA-27: Robotic Arm Controller does not provide "Grasp/suction medicine" during pickup [H8]	UCA-28: Robotic Arm Controller provides "Grasp/suction medicine" in the wrong position [H3]	UCA-29: Robotic Arm Controller provides "Grasp/suction medicine" before positioning [H8]	UCA-30: Robotic Arm Controller initiates "Grasp/suction medicine", but terminates the action before ensuring secure held [H4,H8].
Hold medicine (R-15)	Robotic arm controller/Robotic arm	UCA-31: Robotic Arm Controller does not provide "Hold medicine" after grasping [H4]	UCA-32: Robotic Arm Controller provides "Hold medicine" too tightly [H3] UCA-33: Robotic Arm Controller provides "Hold medicine" inappropriately [H4]	UCA-34: Robotic Arm Controller provides "Hold medicine" too late [H4]	UCA-35: Robotic Arm Controller provides "Hold medicine" but ends before drop point [H4]
Drop medicines (R-21)	Robotic arm/ Medicine bag	UCA-36: Robotic Arm does not execute "Drop medicines" in medicine bag when the arm is correctly positioned [H8]	UCA-37: Robotic Arm executes "Drop medicines" when there is no bag available [H3, H8] UCA-38: Robotic Arm executes "Drop medicines" when it is not positioned directly above the bag [H3, H8]	UCA-39: Robotic Arm executes "Drop medicines" too late, when the bag has already been removed [H3, H8] UCA-40: The robotic arm executed the "Drop Medicines" too late, delivering in a bag intended for another patient [H1c]	Not Applicable

Control Action	From/To	Not Providing	Providing	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Get medicines (R-14)	Robotic arm/ Medicine shelf	UCA-41: Robotic Arm does not "Get medicines" when a dispense command has been issued [H8]	UCA-42: Robotic Arm "Get medicines" without a corresponding valid command [H1, H3, H4, H8] UCA-43: Robotic Arm provides "Get medicines" from incorrect shelf [H1, H3, H4] UCA-44: Robotic Arm provides "Get medicines" of the wrong type (blister/ampoule) [H3, H4, H8] UCA-45: Robotic Arm provides "Get medicines" before item is available [H8]	UCA-46: Robotic Arm "Get medicines" too early, before the correct medication ID or location is confirmed [H1, H3, H4]	UCA-47: Robotic Arm provides "Get medicines" but the action is stopped too soon, preventing medication retrieval [H8]
Authorize operation (R-08, R-11)	Support team/ Pharmacy employee	UCA-48: Support Team does not provide "Authorize operation" when robot is ready [H8]	UCA-49: Support Team provides "Authorize operation" when robot is unsafe [H6]	Not Applicable	Not Applicable
Give support (R-08)	Support team/Pharmacy employee	UCA-50: Support Team does not provide "Give support" when the pharmacy employee does not know what actions to take [H1-H8]	UCA-51: Support team provide "Give support" with erroneous information [H1-H8]	UCA-52: Support Team provides "Give support" too late, after the error happen for long time[H1-H8]	Not Applicable

Source: Author (2025).

Below, a discussion and a table of HCs identified for the robotic arm system during Step 4.8 are presented, following the decision point "Need to model HCs for responsibilities without UCA or add more HCs?".

Additional Hazardous Conditions identified independently of the STPA results

An HC is an element that can address any hazardous situation not captured by a UCA—including those that may hinder the fulfillment of responsibilities.

Table 4.3 presents the Hazardous Conditions (HCs) identified for each responsibility that was not addressed by any UCA. Additionally, there was a case in which a responsibility had not been covered during the UCA identification process, requiring the modeling of a new Hazardous Condition. HC-01 — "Enter the robot's operational area while it is in motion [R-3]" — was modeled as a condition that obstructs the fulfillment of responsibility R-3, assigned to the *Pharmacy Employee* actor. HC-02 — "Unavailability of responsible personnel during emergencies [R-8]" — was deemed necessary, as the analyst identified that this scenario had not been covered through UCA. HC-03 and HC-04 were modeled as hazards that obstruct responsibilities R-5 and R-6, respectively, since these are modeled as feedback mechanisms in the control structure, which prevents them from being framed as UCA. A similar situation

occurs with HC-05, HC-06, and HC-07, which obstruct responsibilities R-16, R-19, and R-20. These responsibilities are also treated through feedback mechanisms in the control structure, and therefore cannot be addressed through UCAs.

Table 4.3: Additional hazardous conditions identified independently of the STPA results

Hazardous Condition	**Description and Related Responsibility**
HC-01	Enter the robot's operational area while it is in motion [R-3]
HC-02	Unavailability of responsible personnel during emergencies [R-8]
HC-03	Do not observe or fail to report abnormal behavior of the robotic arm to the Support Team [R-05]
HC-04	Do not perform periodic checks of the robotic arm's structural integrity or fail to report identified anomalies [R-06]
HC-05	Do not monitor medication possession or fail to signal an anomaly when the medication is lost [R-16]
HC-06	Do not generate operational logs, or generate incomplete logs, preventing anomaly detection and system behavior verification [R-19]
HC-07	Failure to emit alerts upon detecting a malfunction, or emission of incomplete/incorrect alerts [R-20]

Source: Author (2025).

Identify Controller Constraints

Controller constraints specify how a controller must behave or what it must avoid doing to prevent the associated hazards from materializing. These constraints can be simple negations of UCAs or HCs, or they may be more elaborate, depending on the context.

Table 4.4 presents the controller constraints identified in Step 4 for each UCA or Hazardous Condition (HC).

Table 4.4: Controller Constraints for Each Unsafe Control Action (Step 4)

Unsafe Control Action (UCA) or Hazardous Condition (HC)	**Controller Constraint (C)**
UCA-01: Pharmacy Employee does not provide "Check ticket-patient" before starting dispensing [H1]	C-01: The Pharmacy Employee must provide "Check ticket-patient" before starting dispensing [UCA-01]
UCA-02: Pharmacy Employee provides "Check ticket-patient" with incorrect or outdated data [H1]	C-02: The Pharmacy Employee must ensure that the prescription data is current and correct [UCA-02]
UCA-03: Pharmacy Employee provides "Check ticket-patient" too late in the dispensing process [H1]	C-03: The Pharmacy Employee must verify the ticket-patient match before the dispensing process begins [UCA-03]
UCA-04: Pharmacy Employee provides "Check ticket-patient" after the medication has already been dispensed [H1]	C-04: The Pharmacy Employee must perform the check before dispensing [UCA-04]
UCA-05: Pharmacy Employee does not provide "Restart ticket dispensing" after failure to complete dispensing [H8]	C-05: The Pharmacy Employee must issue a restart command if the dispensing was not completed [UCA-05]

Unsafe Control Action (UCA) or Hazardous Condition (HC)	Controller Constraint (C)
UCA-06: Pharmacy Employee provides "Restart ticket dispensing" unnecessarily [H1]	C-06: The Pharmacy Employee must only restart dispensing when necessary. [UCA-06]
UCA-07: Pharmacy Employee provides "Restart ticket dispensing" and the robotic arm is not prepared yet [H8]	C-07: The Pharmacy Employee must confirm robotic arm readiness before restarting dispensing. [UCA-07]
UCA-08: Pharmacy Employee does not provide "Check bag-patient" before dispensing [H1c]	C-08: The Pharmacy Employee must check bag-patient association before dispensing. [UCA-08]
UCA-09: Pharmacy Employee provides "Check bag-patient" with wrong assignment confirmation [H1c]	C-09: The Pharmacy Employee must ensure correct assignment when checking bag-patient. [UCA-09]
UCA-10: Support Team does not provide "Diagnose robotic arm failures" after detecting abnormal behavior [H1-H8]	C-10: The Support Team must diagnose robotic failures upon abnormal behavior. [UCA-10]
UCA-11: Support Team provides "Diagnose robotic arm failures" with wrong conclusions [H1-H8]	C-11: The Support Team must ensure accurate conclusions when diagnosing robotic failures. [UCA-11]
UCA-12: Support Team provides "Diagnose robotic arm failures" too late [H1-H8]	C-12: The Support Team must promptly diagnose failures. [UCA-12]
UCA-13: Support Team provides "Diagnose robotic arm failures" but stops before completing the analysis [H1-H8]	C-13: The Support Team must complete the full diagnostic analysis. [UCA-13]
UCA-14: Support Team does not provide "Interrupt robot operation" when the robot is malfunctioning [H1-H8]	C-14: The Support Team must interrupt operation when malfunction is detected. [UCA-14]
UCA-15: Support Team provides "Interrupt robot operation" when it is not necessary [H8]	C-15: The Support Team must avoid interrupting robot operation without need. [UCA-15]
UCA-16: Support Team provides "Interrupt robot operation" too late [H1-H8]	C-16: The Support Team must promptly interrupt robot operation when necessary. [UCA-16]
UCA-17: Support Team does not provide "Perform preventive inspection" as scheduled [H5, H7]	C-17: The Support Team must follow the preventive inspection schedule. [UCA-17]
UCA-18: Support Team does not provide "Verify structural integrity" of robotic arm when it is necessary[H5, H7]	C-18: The Support Team must verify the structural integrity when required. [UCA-18]
UCA-19: Support Team provides "Verify structural integrity" using incomplete or invalid data [H7, H8]	C-19: The Support Team must use valid and complete data to verify structural integrity. [UCA-19]
UCA-20: Robotic Arm Controller does not provide "Check medication tag" before approaching the medication for grasping or suction [H1, H8]	C-20: The Robotic Arm must perform "Check medication tag" before approaching the medication. [UCA-20]
UCA-21: Robotic Arm Controller performs "Check medication tag", but the tag is associated with the wrong item [H1]	C-21: The Robotic Arm must ensure the tag is correctly associated with the intended medication. [UCA-21]
UCA-22: Robotic Arm Controller performs "Check medication tag" prematurely, resulting in the detection of the wrong medication [H1]	C-22: The Robotic Arm must perform "Check medication tag" at the appropriate moment. [UCA-22]
UCA-23: Robotic Arm Controller performs "Check medication tag" too late, causing the medication not to be detected [H8]	C-23: The Robotic Arm must check the medication tag early enough to ensure detection. [UCA-23]
UCA-24: Robotic Arm Controller provides "Check medication tag" but ends before full read [H8]	C-24: The Robotic Arm must complete the medication tag reading process. [UCA-24]
UCA-25: Robotic Arm Controller does not provide "Control movements" while the robotic arm is in motion [H2, H5, H6, H7, H8]	C-25: The Robotic Arm must receive continuous control commands during movement. [UCA-25]
UCA-26: Robotic Arm Controller provides "Control movements" in incorrect sequence [H4, H6, H8]	C-26: The Robotic Arm must execute movements in the correct sequence. [UCA-26]
UCA-27: Robotic Arm Controller does not provide "Grasp/suction medicine" during pickup [H8]	C-27: The Robotic Arm must perform the grasp action during pickup. [UCA-27]
UCA-28: Robotic Arm Controller provides "Grasp/suction medicine" in the wrong position [H3]	C-28: The Robotic Arm must be positioned correctly before grasping. [UCA-28]
UCA-29: Robotic Arm Controller provides "Grasp/suction medicine" before positioning [H8]	C-29: The Robotic Arm must position itself before initiating grasping. [UCA-29]
UCA-30: Robotic Arm Controller initiates "Grasp/suction medicine", but terminates the action before ensuring secure held [H4,H8].	C-30: The Robotic Arm must complete suction action to ensure secure medication grasp. [UCA-30]
UCA-31: Robotic Arm Controller does not provide "Hold medicine" after grasping [H4]	C-31: The Robotic Arm must maintain a hold on the medication after grasping. [UCA-31]
UCA-32: Robotic Arm Controller provides "Hold medicine" too tightly [H3]	C-32: The Robotic Arm must control the gripping force to avoid damaging the medication. [UCA-32]

Unsafe Control Action (UCA) or Hazardous Condition (HC)	Controller Constraint (C)
UCA-33: Robotic Arm Controller provides “Hold medicine” inappropriately [H4]	C-33: The Robotic Arm must apply appropriate holding force. [UCA-33]
UCA-34: Robotic Arm Controller provides “Hold medicine” too late [H4]	C-34: The Robotic Arm must hold the medication immediately after pickup. [UCA-34]
UCA-35: Robotic Arm Controller provides “Hold medicine” but ends before drop point [H4]	C-35: The Robotic Arm must maintain hold on the medication until reaching the drop point. [UCA-35]
UCA-36: Robotic Arm does not execute “Drop medicines” in medicine bag when the arm is correctly positioned [H8]	C-36: The Robotic Arm must drop the medicine only when above the correct bag. [UCA-36]
UCA-37: Robotic Arm executes “Drop medicines” when there is no bag available [H3, H8]	C-37: The Robotic Arm must ensure the presence of a bag before dropping medication. [UCA-37]
UCA-38: Robotic Arm executes “Drop medicines” when it is not positioned directly above the bag [H3, H8].	C-38: The Robotic Arm must verify alignment above the bag before releasing. [UCA-38]
UCA-39: Robotic Arm executes “Drop medicines” too late, when the bag has already been removed [H3, H8]	C-39: The Robotic Arm must drop the medicine before the bag is removed. [UCA-39]
UCA-40: The robotic arm executed the “Drop Medicines” too late, delivering in a bag intended for another patient [H1c]	C-40: The Robotic Arm must ensure the correct bag before dropping. [UCA-40]
UCA-41: Robotic Arm does not “Get medicines” when a dispense command has been issued [H8]	C-41: The Robotic Arm must execute the get command upon valid instruction. [UCA-41]
UCA-42: Robotic Arm “Get medicines” without a corresponding valid command [H1, H3, H4, H8]	C-42: The Robotic Arm must only get medicines after a valid command. [UCA-42]
UCA-43: Robotic Arm provides “Get medicines” from incorrect shelf [H1, H3, H4]	C-43: The Robotic Arm must verify the correct shelf before retrieving. [UCA-43]
UCA-44: Robotic Arm provides “Get medicines” of the wrong type (blister/ampoule) [H3, H4, H8]	C-44: The Robotic Arm must confirm the medicine type before retrieving. [UCA-44]
UCA-45: Robotic Arm provides “Get medicines” before item is available [H8]	C-45: The Robotic Arm must check for availability before retrieving. [UCA-45]
UCA-46: Robotic Arm “Get medicines” too early, before the correct medication ID or location is confirmed [H1, H3, H4]	C-46: The Robotic Arm must verify medication ID and location before initiating pickup. [UCA-46]
UCA-47: Robotic Arm provides “Get medicines” but the action is stopped too soon, preventing medication retrieval [H8]	C-47: The Robotic Arm must complete the get action before stopping. [UCA-47]
UCA-48: Support Team does not provide “Authorize operation” when robot is ready [H8]	C-48: The Support Team must authorize operation when all conditions are met. [UCA-48]
UCA-49: Support Team provides “Authorize operation” when robot is unsafe [H6]	C-49: The Support Team must ensure safety before authorizing operation. [UCA-49]
UCA-50: Support Team does not provide “Give support” when the pharmacy employee does not know what actions to take [H1-H8]	C-50: The Support Team must support pharmacy staff when guidance is needed. [UCA-50]
UCA-51: Support team provide “Give support” with erroneous information [H1-H8]	C-51: The Support Team must ensure correctness of the information provided. [UCA-51]
UCA-52: Support Team provides “Give support” too late, after the error happen for long time[H1-H8]	C-52: The Support Team must respond promptly to detected errors. [UCA-52]
HC-01: Enter the robot’s operational area while it is in motion [R-3]	C-53:The robot’s operational area must not be accessed while the robot is in motion. [HC-01]
HC-02: Unavailability of responsible personnel during emergencies. [R-8]	C:54: Responsible personnel must always be available during emergencies. [HC-02]
HC-03: Do not observe or fail to report abnormal behavior of the robotic arm to the Support Team. [R-05]	C-55: The Support Team must be notified immediately upon detecting any abnormal robotic arm behavior. [HC-03]
HC-04: Do not perform periodic checks of the robotic arm’s structural integrity or fail to report identified anomalies.[R-06]	C-56: Periodic checks of the robotic arm’s integrity must be conducted and any anomalies must be promptly reported. [HC-04]
HC-05: Do not monitor medication possession or fail to signal an anomaly when the medication is lost. [R-16]	C-57: Medication possession must be continuously monitored, and any anomalies must be signaled immediately. [HC-05]
HC-06: Do not generate operational logs, or generate incomplete logs, preventing anomaly detection and system behavior verification. [R-19]	C-58: The system must generate complete and accurate operational logs to support anomaly detection and behavior verification. [HC-06]
HC-07: Failure to emit alerts upon detecting a malfunction, or emission of incomplete/incorrect alerts. [R-20]	C-59: The system must emit complete and accurate alerts immediately upon detecting any malfunction. [HC-07]

Source: Author (2025).

Having defined the constraints, the analyst should now proceed to Step 5, where loss scenarios and safety requirements must be defined for each identified UCA. This step is explained and applied in the following section.

4.6.5 Step 5 – Analyze Loss Scenarios and Derive Safety Requirements

We now present the loss scenarios and safety requirements defined during the analysis for each identified UCA and HC found in Step 4. Loss scenarios are specific combinations of conditions, environmental factors, or operational errors that could cause the selected UCA or HC to result in a system-level hazard. Each potential path leading from a UCA or HC to a hazard is documented as a unique loss scenario. Safety requirements, in turn, are requirements designed to prevent, mitigate, or detect the identified loss scenarios. Each safety requirement is derived with the objective of eliminating the causal factors or conditions that could allow the loss to occur. These safety requirements are recorded in a dedicated artifact and become an integral part of the overall safety specification of the system.

In summary, LSs are generated for one or more identified UCAs or HCs and must be associated with them. Likewise, SRs are generated for one or more LSs and must also be linked accordingly.

Table 4.5 presents the LSs and their corresponding SR for each UCA and HC defined for the robotic arm medication delivery system.

Table 4.5: Loss Scenarios and Safety Requirements associated with UCAs

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-01: Pharmacy Employee does not provide "Check ticket-patient" before starting dispensing [H1]	LS-01: Pharmacy Employee is, by design, responsible for always performing "Check ticket-patient" during every dispensing session, even if the system allows dispensing to proceed without verification. [UCA-01] LS-02: Pharmacy Employee incorrectly believes that the ticket is in a status other than "Waiting", due to a status update error, and therefore skips the ticket-patient verification. [UCA-01] LS-03: Pharmacy Employee's Process Model is updated with a status "In Progress", even though ticket dispensing should have been restarted, leading to skipped verification. [UCA-01] LS-04: If the ticket status feedback "Waiting" is not defined, then the interface may fail to display the "Check ticket-patient" option. [UCA-01] LS-05: Pharmacy Employee does not prevent skipping the ticket-patient check if the printed ticket incorrectly indicates that the check has already been performed. [UCA-01] LS-06: Robotic Arm Controller does not block the dispensing command when the ticket has not been verified, and as a result, Pharmacy Employee neglects the check. [UCA-01]	SR-01: The user interface must always display the mandatory "Check ticket-patient" option when the ticket status is "Waiting". [LS-01, LS-04, LS-05, LS-06] SR-02: The ticket status must be updated and displayed in the ticket selection application interface, and the system must cross-check the database for the current status before allowing dispensing to begin. [LS-02] SR-03: If dispensing needs to be restarted, the Pharmacy Employee's Process Model must be updated with the status "Waiting.". [LS-03] SR-04: If dispensing needs to be restarted, the Pharmacy Employee must verify whether any medication has already been dispensed and remove it from the dispensing bag to allow for correct process reinitialization. [LS-03] SR-05: The initial status of any ticket prepared for dispensing must be "Waiting". [LS-04]
UCA-02: Pharmacy Employee provides "Check ticket-patient" with incorrect or outdated data [H1]	LS-07: The Pharmacy Employee chose to update the online ticket using information from the printed ticket, resulting in outdated data being propagated. [UCA-02] LS-08: The Pharmacy Employee restarted ticket delivery, but in the meantime, the ticket was updated online by the responsible physician, and the update was not reflected in the new delivery action. [UCA-02] LS-09: The Pharmacy Employee believes that the ticket is up to date and correct because the process model does not reflect ticket updates made by the physician in the AGHU system. [UCA-02] LS-10: The Pharmacy Employee does not receive information that the ticket has been updated before checking it and initiating a new dispensing action. [UCA-02] LS-11: If the Pharmacy Employee is under high workload, they may fail to notice that the ticket has been updated. [UCA-02]	SR-06: Only authorized users shall be allowed to update ticket information. [LS-07] SR-07: The robotic arm system shall display the date, time, and user responsible for the last ticket update. [LS-07] SR-08: The Pharmacy Employee shall always verify that the printed ticket matches the online ticket data. [LS-07, LS-08] SR-09: The Pharmacy Employee shall always print a new ticket if the printed version does not match the online ticket, discarding the outdated printed version. [LS-07] SR-10: The robotic arm system shall trigger a synchronization command with the AGHU system to update ticket information every time it is accessed. [LS-08, LS-09, LS-10, LS-11]
UCA-03: Pharmacy Employee provides "Check ticket-patient" too late in the dispensing process [H1]	Mitigated through software lockout, see SR-01	(REPEATED) SR-01: The user interface must always display the mandatory "Check ticket-patient" option when the ticket status is "Waiting". [LS-01, LS-04, LS-05, LS-06]
UCA-04: Pharmacy Employee provides "Check ticket-patient" after the medication has already been dispensed [H1]	LS-12: If the Pharmacy Employee accesses the ticket after it has already been dispensed, they may perform "Check ticket-patient" unnecessarily. [UCA-04]	SR-11: The robotic arm system shall allow the "Check ticket-patient" option only when the ticket is in the "Waiting" state. [LS-12]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-05: Pharmacy Employee does not provide "Restart ticket dispensing" after failure to complete dispensing [H8]	LS-13: The pharmacy employee does not realize that the ticket is in an "Error" status or is unaware of the need to restart the ticket delivery due to lack of training. [UCA-05] LS-14: The system issued an error status, but the Pharmacy Employee ignored it due to cognitive overload or competing tasks. [UCA-05] LS-15: The Pharmacy Employee assumes the dispensing process completed successfully due to absence of an audible or visual alert from the system. [UCA-05] LS-16: The robotic arm returned to its idle state without indicating that the dispensing process had failed. [UCA-05] LS-17: The Pharmacy Employee's process model was not updated with the "Error" status due to delayed or failed communication from the dispensing system. [UCA-05] LS-18: The Pharmacy Employee assumed that a next dispensing would be automatically triggered by the system, and thus did not initiate it manually. [UCA-05] LS-19: During a robotic system malfunction, the system may incorrectly label tickets as "Error" despite the issue not affecting the ability to complete their delivery. [UCA-05]	SR-12: The pharmacy employee's training must include guidance on the need to restart the ticket whenever required. [LS-13, LS-18] SR-13: The robotic system must generate a clear and unambiguous alert (visual and audible) when the ticket status is set to "Error". [LS-14, LS-15] SR-14: The robotic arm must explicitly report the dispensing result (success or failure) before returning to idle. [LS-16] SR-15: The ticket status in the PE's PM must be updated in real-time when the system reports a failed dispensing status. [LS-17] SR-16: The system must distinguish between robot errors and ticket status, ensuring that medication delivery is not blocked unless strictly necessary. [LS-19]
UCA-06: Pharmacy Employee provides "Restart ticket dispensing" unnecessarily [H1]	LS-20: The pharmacy employee assumes the medicine bag holds an incorrect amount of medication as a result of relying on an outdated ticket. [UCA-06] LS-21: The user interface shows ambiguous or delayed feedback, leading the Pharmacy Employee to assume failure. [UCA-06] LS-22: The interface allows restarting even when the ticket status is "Completed," without requesting confirmation. [UCA-06] LS-23: A network delay caused a status mismatch between the online system and the interface available to the PE. [UCA-06] LS-24: In the absence of ticket status information, the pharmacy employee may assume that restarting the dispensing process is required. [UCA-06]	SR-17: Restart functionality must require validation against the current ticket status stored in the central database. [LS-20] SR-18: The system must disable the "Restart ticket dispensing" function when the ticket status is "Completed". [LS-21, LS-22] SR-19: The interface must clearly indicate the dispensing completion status with timestamp and delivery log. [LS-21, LS-23] SR-20: When ticket status information is unavailable, the pharmacy employee must verify the status manually and respond based on their training. [LS-24] .

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-07: Pharmacy Employee provides "Restart ticket dispensing" and the robotic arm is not prepared yet [H8]	LS-25: The Pharmacy Employee assumes the robotic arm is ready based on its physical position or silence, without confirming its internal state. [UCA-07] LS-26: The robotic arm may accept commands to restart dispensing even though it is not yet in a state ready to initiate the dispensing process. [UCA-07] LS-27: The system interface fails to display or update the robotic arm readiness status in real time. [UCA-07] LS-28: The Pharmacy Employee is unaware of a pending calibration or error recovery operation in progress. [UCA-07] LS-29: The restart command is executed while the robotic arm is physically obstructed or disabled, risking a mechanical hazard. [UCA-07] LS-30: Due to lack of training, the pharmacy employee interacts solely through the interface and is not prepared to handle the physical robot or interpret its behavior. [UCA-07]	SR-21: The interface must visibly and clearly indicate the robotic arm readiness status at all times. [LS-25] SR-22: The pharmacy employee must, based on their training, check essential conditions to confirm that the robot is functioning correctly. [LS-25] SR-23: The robotic arm system must reject restart commands while it is not in a ready state. [LS-25, LS-26, LS-29, LS-30] SR-24: The interface must visibly and clearly indicate the robotic arm readiness status at all times. [LS-26] SR-25: Restart buttons must be disabled until the robotic system initialization and safety checks are completed. [LS-26] SR-26: A warning must be issued if the PE attempts to restart dispensing while the robotic arm is offline, calibrating, or under error state. [LS-26, LS-28, LS-29, LS-30] SR-27: If communication with the robotic arm fails, the restart function must be blocked and flagged as unsafe. [LS-27]
UCA-08: Pharmacy Employee does not provide "Check bag-patient" before dispensing [H1c]	LS-31: The Pharmacy Employee skips the bag-patient check assuming the bag is already positioned correctly for the patient. [UCA-08] LS-32: The user interface allows dispensing without requiring explicit confirmation of the bag-patient match. [UCA-08]	SR-27: The system must enforce explicit confirmation of bag-patient verification before enabling dispensing. [LS-31] SR-28: The system must generate a warning if dispensing is attempted before verifying bag-patient correspondence. [LS-31] SR-29: The interface must highlight and require the "Check bag-patient" action when a bag is detected in position. [LS-32]
UCA-09: Pharmacy Employee provides "Check bag-patient" with wrong assignment confirmation [H1c]	LS-33: The user interface displays outdated or incorrect bag assignment due to delay in status synchronization. [UCA-09] LS-34: The pharmacy employee does not confirm that the bag has been replaced and continues using the previous medicine bag, placing medications intended for another patient. [UCA-09] LS-35: The pharmacy employee does not update the bag after an error occurs and restarts the dispensing process, resulting in the wrong quantity of medication being added. [UCA-09] LS-36: The bag was moved after its initial assignment, but the Pharmacy Employee is unaware of the reassignment. [UCA-09]	SR-30: The interface must display real-time, clearly visible assignment data, including patient name and bag code. [LS-33] SR-31: The pharmacy employee must physically verify the medicine bag before restarting or initiating any ticket delivery. [LS-34, LS-35] SR-32: The pharmacy employee must physically verify the position of the medicine bag before restarting or initiating any delivery. [LS-36]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-10: Support Team does not provide "Diagnose robotic arm failures" after detecting abnormal behavior [H1-H8]	LS-37: The Support Team is notified of the anomaly but postpones diagnosis due to workload or lack of prioritization. [UCA-10]	SR-33: All robot behavior anomalies must be logged and immediately reported to the Support Team interface. [LS-38, LS-39, LS-40, LS-42, LS-43]
	LS-38: The Support Team receives a failure alert, but it is misclassified or marked as a minor issue in the system. [UCA-10]	SR-34: The system must require formal acknowledgment and diagnosis of each flagged anomaly before normal operation can resume. [LS-37–LS-43]
	LS-39: The abnormal behavior was logged but not communicated in real time to the Support Team interface. [UCA-10]	SR-35: Diagnostic actions must be prioritized in the interface and visually distinguished from other tasks. [LS-42, LS-43]
	LS-40: The Support Team assumes the behavior is a known transient issue and does not investigate further. [UCA-10]	SR-36: The Support Team must receive real-time access to logs and system state snapshots when a failure is detected. [LS-41, LS-42]
	LS-41: The diagnostic tools are offline or unavailable when the failure occurs, delaying the response. [UCA-10]	SR-37: Robot anomalies must be cross-referenced with a severity matrix to enforce mandatory investigation. [LS-42]
	LS-42: The failure log lacked sufficient context or data to initiate diagnosis, and no protocol enforced further action. [UCA-10]	SR-38: If the robot resumes normal behavior after an anomaly, the system must still enforce a diagnostic check before future tasks. [LS-43]
	LS-43: The robot resumed normal operation after the anomaly, misleading the Support Team into thinking no failure occurred. [UCA-10]	
UCA-11: Support Team provides "Diagnose robotic arm failures" with wrong conclusions [H1-H8]	LS-44: The Support Team interprets a transient sensor failure as resolved, when the underlying mechanical issue persists. [UCA-11]	SR-39: All robot failure diagnoses must be validated against current behavioral data before resolution. [LS-45]
	LS-45: The diagnostic log is incomplete, omitting critical data from the time of the anomaly, leading to incorrect reasoning. [UCA-11]	SR-40: The Support Team interface must require review of complete logs and robot status prior to closing a diagnosis. [LS-45, LS-47]
	LS-46: The team misclassifies the type of anomaly due to similar past cases that had different causes. [UCA-11]	SR-41: When abnormal behavior continues after a diagnosis, the system must trigger a re-escalation. [LS-44]
	LS-47: Interface layout or tool limitations prevent the team from accessing deeper robot telemetry necessary for accurate analysis. [UCA-11]	SR-42: Structured diagnostic protocols must guide the Support Team through verification steps before issuing conclusions. [LS-44, LS-45, LS-51]
	LS-48: The system erroneously flags the robot as "normal" due to reset commands, while the failure state remains. [UCA-11]	SR-43: Any diagnosis relying on incomplete or past data must be flagged as preliminary and reviewed by a secondary operator. [LS-46, LS-48, LS-49]
	LS-49: High workload or external pressure causes the team to prematurely validate the robot as functioning. [UCA-11]	SR-44: The system shall implement an alarm prioritization mechanism to guide the Support Team toward the most critical failures first. [LS-50]
	LS-50: Multiple alarms were active, and the team focused on the wrong signal or failure condition. [UCA-11]	SR-45: Training and decision-support documentation shall be embedded in the diagnostic interface to help the Support Team interpret concurrent alarms accurately. [LS-50]
	LS-51: Confirmation bias leads the team to rule out failure because no prior issue of this type has occurred. [UCA-11]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-12: Support Team provides "Diagnose robotic arm failures" too late [H1-H8]	LS-52: The Support Team receives the failure notification but delays the response due to competing maintenance requests. [UCA-12]	SR-46: The Support Team must be trained to respond to robotic system alerts over primary and secondary sources (e.g., verbal reports). [LS-53, LS-54, LS-57, LS-58]
	LS-53: The failure alert is displayed in a low-priority section of the interface, and the operator overlooks it. [UCA-12]	SR-47: The interface must clearly distinguish unresolved critical failures from background system status messages. [LS-52, LS-53, LS-56]
	LS-54: The Support Team assumed the issue was minor and expected it to self-resolve without intervention. [UCA-12]	SR-48: Diagnostic tools must be accessible and functional at all times; fallback mechanisms must exist during maintenance windows. [LS-52, LS-53, LS-56]
	LS-55: The system generated the alert during a shift change or unstaffed period, causing delay in response. [UCA-12]	SR-49: Alerts must be accompanied by audible signals and color-coded interface indicators until explicitly acknowledged. [LS-53, LS-54, LS-56]
	LS-56: The system failed to trigger an audible or prominent alert, so the Support Team did not notice the failure promptly. [UCA-12]	SR-50: A dashboard must display pending failure conditions in order of criticality to direct attention efficiently. [LS-55]
	LS-57: Diagnostic tools required for analysis were temporarily unavailable or under maintenance. [UCA-12]	
	LS-58: The Support Team relied on PE verbal reports instead of the robotic arm's system logs, delaying proper assessment. [UCA-12]	
UCA-13: Support Team provides "Diagnose robotic arm failures" but stops before completing the analysis [H1-H8]	LS-59: The Support Team exits the diagnostic screen after checking initial parameters but before completing the full workflow. [UCA-13]	SR-51: The diagnostic process must not be considered complete unless all critical steps are confirmed and validated. [LS-59, LS-61]
	LS-60: An unrelated alert or task interrupts the diagnostic session, and the team does not resume the analysis. [UCA-13]	SR-52: The system must issue a warning if the Support Team exits diagnostics before reaching final verification. [LS-59, LS-62]
	LS-61: The diagnostic checklist was not fully completed, but the system allowed the session to be closed. [UCA-13]	SR-53: The interface must display a checklist with clear indicators of pending diagnostic steps. [LS-61, LS-63]
	LS-62: The Support Team believed the issue was resolved based on partial information and prematurely ended the diagnosis. [UCA-13]	SR-54: The system must prevent closure of the diagnostic session until root cause analysis or final observation is entered. [LS-64, LS-66]
	LS-63: The interface did not indicate which steps were still pending in the diagnostic procedure. [UCA-13]	SR-55: If a diagnostic session is interrupted (e.g., due to alert or crash), the system must prompt automatic resumption or revalidation upon return. [LS-60, LS-65]
	LS-64: The system did not flag the absence of a final validation or root cause identification before the diagnosis was closed. [UCA-13]	SR-56: Diagnostic results must be tagged with a completion status flag and blocked from being used for safety clearance if marked as incomplete. [LS-66]
	LS-65: The diagnostic tools malfunctioned mid-session, and no automatic mechanism prompted continuation after recovery. [UCA-13]	SR-57: The robotic system must maintain restricted operation mode if diagnostics are incomplete and the robot is still in an abnormal state. [LS-64]
	LS-66: The diagnostic report was saved automatically even though not all fields or checks were completed. [UCA-13]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-14: Support Team does not provide "Interrupt robot operation" when the robot is malfunctioning [H1-H8]	LS-67: The Support Team observes the malfunction but assumes it is non-critical and chooses not to interrupt. [UCA-14]	SR-58: The system must provide Support Team personnel with the authority and interface access to immediately interrupt robotic operations. [LS-68, LS-69]
	LS-68: The malfunction is confirmed in the system, but the Support Team lacks access permissions to issue the interrupt command. [UCA-14]	SR-59: A confirmed robot malfunction must trigger an interrupt alert requiring acknowledgment or immediate action. [LS-67, LS-70]
	LS-69: The interface for interrupting operation is non-responsive or unclear during the failure event. [UCA-14]	SR-60: The interface must clearly highlight malfunction conditions and guide the user to the interrupt procedure. [LS-69, LS-71]
	LS-70: Multiple failures are occurring simultaneously, and the Support Team focuses on another subsystem. [UCA-14]	SR-61: Interrupt commands must bypass non-essential confirmation steps during active failure states. [LS-68, LS-73]
	RS-71: A system bug delays the visualization of the failure flag on the Support Team's dashboard. [UCA-14]	SR-62: A protocol must define failure severity thresholds and mandatory interruption conditions. [RS-70, RS-74]
	LS-72: The robotic system incorrectly displays a normal state, despite logged failure behavior. [UCA-14]	SR-63: The system must display a discrepancy alert if behavior logs indicate failure while the robot reports a normal state. [LS-72]
	LS-73: The Support Team hesitates to interrupt due to fear of disrupting patient-facing processes without confirmation. [UCA-14]	SR-64: A redundant fail-safe interrupt option must be available if the primary command fails. [LS-69]
	LS-74: There is no defined protocol indicating when the robot must be forcefully interrupted, leading to indecision. [UCA-14]	SR-65: Interrupt decisions must be logged with timestamps, operator ID, and justification for audit and future analysis. [LS-67, LS-68]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-15: Support Team provides "Interrupt robot operation" when it is not necessary [H8]	LS-75: A transient sensor fluctuation is misclassified by the team as an operational hazard. [UCA-15]	SR-66: The system must verify real-time operational readiness before accepting an interrupt command. [LS-77]
	LS-76: The robot is momentarily idle between tasks, and the Support Team incorrectly assumes it is malfunctioning. [UCA-15]	SR-67: The interface must show confirmation of "normal operation" explicitly and update continuously. [LS-77, LS-79]
	LS-77: The interface displays outdated or lagged data, misleading the Support Team into thinking a problem persists. [UCA-15]	SR-68: All interrupt requests must be justified by a logged failure condition or override protocol. [LS-81]
	LS-78: The Support Team applies a standard protocol for interruption without reviewing the current system state. [UCA-15]	SR-69: Alerts must distinguish between critical failures and informational notifications. [LS-75, LS-79]
	LS-79: A lack of feedback clarity results in a false positive visual alert prompting an unnecessary action. [UCA-15]	SR-70: The Support Team must confirm current telemetry data before taking interrupt action. [LS-76, LS-80]
	LS-80: Communication delays cause telemetry inconsistencies, appearing as faults. [UCA-15]	SR-71: If no failure is detected, the system must prompt the operator to review the status before proceeding with interruption. [LS-78]
	LS-81: The Support Team interrupts the operation based on a user complaint without technical verification. [UCA-15]	SR-72: Interrupt logs must include system state at the time of action and reasoning for operator decision. [LS-81]
UCA-16: Support Team provides "Interrupt robot operation" too late [H1-H8]	LS-82: The Support Team delays issuing the interrupt command while awaiting additional confirmation or logs. [UCA-16]	SR-73: The system must monitor the duration of any critical failure state and send additional alert after a critical threshold. [LS-82, LS-84]
	LS-83: The failure alert was acknowledged but not acted upon due to competing tasks o cognitive overload. [UCA-16]	SR-74: All critical alerts must remain visible and prioritized on the interface until acknowledged and resolved. [LS-83, LS-85]
	LS-84: There is no clear threshold defined in the system for when interruption becomes mandatory. [UCA-16]	SR-75: Interruption must be enforced when the system detects persistent failure beyond acceptable timeframes. [LS-82, LS-84, LS-88]
	LS-85: The alert system failed to maintain visibility after initial acknowledgment, causing loss of attention. [UCA-16]	SR-76: If the interrupt is not performed within the critical window, the system must either auto-interrupt or block new operations. [LS-84, LS-86]
	LS-86: The Support Team lacked access to perform the interrupt at the time of the alert. [UCA-16]	SR-77: The interface must provide clear escalation cues (e.g., color, sound, countdowns) for delayed interrupt actions. [LS-85, LS-87]
	LS-87: The interface marked the issue as low priority or pending diagnostics, causing delay in action. [UCA-16]	SR-78: The Support Team must have 24/7 access to the interrupt function or a fallback escalation contact. [LS-86]
	LS-88: The Support Team assumed the malfunction would self-resolve and opted to wait. [UCA-16]	SR-79: Every delay in interruption beyond threshold must be logged with timestamps, reasoning, and responsible personnel. [LS-82, LS-83]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-17: Support Team does not provide "Perform preventive inspection" as scheduled [H5, H7]	LS-89: The Support Team overlooks the inspection notification due to poor visibility or placement in the interface. [UCA-17]	SR-80: The system must display scheduled preventive inspections prominently and issue countdown alerts. [LS-89, LS-90]
	LS-90: The system failed to issue the notification on time due to a scheduling bug. [UCA-17]	SR-81: Preventive inspections must be tracked with assigned responsibility and completion status. [LS-91, LS-93]
	LS-91: The preventive inspection alert was acknowledged but postponed indefinitely due to higher-priority issues. [UCA-17]	SR-82: The interface must distinguish preventive inspection alerts from other notifications using specific icons, colors, or categories. [LS-89, LS-94]
	LS-92: The Support Team is unaware of the inspection schedule due to lack of documentation or calendar integration. [UCA-17]	SR-83: If the inspection deadline passes without confirmation, the system must block robot operation or flag the condition as unsafe. [LS-91, LS-95]
	LS-93: Preventive inspection tasks are not assigned to specific team members, resulting in confusion or omission. [UCA-17]	SR-84: The Support Team must be able to access an inspection history log and upcoming schedule directly from the dashboard. [LS-92, LS-93]
	LS-94: The interface does not differentiate between preventive inspection alerts and general notifications. [UCA-17]	SR-85: A preventive inspection must not be overridden by recent corrective maintenance unless explicitly authorized and documented. [LS-96]
	LS-95: No consequence is enforced when the inspection is missed, so the team deprioritizes the task. [UCA-17]	SR-86: Missed or delayed inspections must be logged with timestamps, responsible personnel, and justification. [LS-91, LS-95]
	LS-96: The Support Team believes recent corrective maintenance replaced the need for preventive inspection. [UCA-17]	SR-87: The system must escalate overdue inspections by notifying supervisors or maintenance leads. [LS-91, LS-92]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-18: Support Team does not provide "Verify structural integrity" of robotic arm when it is necessary[H5, H7]	LS-97: The system detects abnormal stress levels or vibrations but does not notify the Support Team. [UCA-18]	SR-88: The system must issue a high-priority alert when anomalies in structural parameters (e.g., torque variance, deformation) are detected. [LS-97, LS-103]
	LS-98: The Pharmacy Employee observes abnormal movement or noise but does not report it to the Support Team. [UCA-18]	SR-89: Structural warnings must include severity level, timestamp, and sensor data to guide the Support Team's response. [LS-97, LS-102]
	LS-99: Visual or auditory signs of wear (e.g., shaking, misalignment) are misinterpreted as normal behavior by the Pharmacy Employee. [UCA-18]	SR-90: Training must be provided to Pharmacy Employees to recognize and report symptoms of structural wear or failure. [LS-98, LS-99]
	LS-100: The interface shows a general alert but fails to specify that a structural inspection is needed. [UCA-18]	SR-91: The interface must clearly label alerts that require structural integrity verification and separate them from informational messages. [LS-100, LS-101]
	LS-101: The alert appears in a secondary panel or tab, reducing its visibility to the Support Team. [UCA-18]	SR-92: Alerts regarding structural checks must be persistent until acknowledged and resolved by the Support Team. [LS-100, LS-102]
	LS-102: The Support Team ignores the alert assuming it is a non-critical advisory instead of a structural warning. [UCA-18]	SR-93: All reported symptoms and inspection requests must be logged and traced back to the triggering observation (system or human). [LS-98]
	LS-103: The robot's structural integrity status is outdated due to delayed synchronization between sensor input and display. [UCA-18]	SR-94: The robot must enter restricted or monitored mode if structural verification is overdue. [LS-102]
UCA-19: Support Team provides "Verify structural integrity" using incomplete or invalid data [H7, H8]	LS-104: The Support Team analyzes only the last captured data frame, ignoring time-series trends or prior anomalies. [UCA-19]	SR-95: The system must alert the Support Team when structural integrity data is incomplete or partially missing. [LS-105, LS-107]
	LS-105: The structural integrity report is missing logs from specific joints due to a temporary sensor failure. [UCA-19]	SR-96: All structural inspection reports must include full telemetry history, with time synchronization and data source tags. [LS-104, LS-106, LS-109]
	LS-106: The inspection is performed while telemetry is desynchronized, leading to mismatched data intervals. [UCA-19]	SR-97: The interface must block structural verification if any required sensor input or timestamped log is absent or inconsistent. [LS-105, LS-107, LS-110]
	LS-107: The system interface allows the inspection to proceed without warning about missing or outdated data. [UCA-19]	SR-98: A warning must be issued if old or cached data is used for verification without confirmation of freshness. [LS-108]
	LS-108: Partial inspection data is cached, and a previously saved report is used without verifying freshness. [UCA-19]	SR-99: If data is compressed or summarized visually, the interface must offer an option to expand and inspect the full raw data. [LS-110]
	LS-109: Minor sensor inconsistencies (e.g., jitter, noise) are not filtered or flagged, causing false-positive conclusions. [UCA-19]	SR-100: Any inspection based on degraded data must be labeled as provisional and require secondary validation. [LS-104, LS-106]
	LS-110: The interface compresses the report or omits some low-severity indicators due to layout limitations. [UCA-19]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-20: Robotic Arm Controller does not provide "Check medication tag" before approaching the medication for grasping or suction [H1, H8]	LS-111: The Robotic Arm Controller receives valid position feedback but skips the tag verification due to software flow bypass. [UCA-20]	SR-101: Tag verification must be explicitly linked to arm position feedback and blocked if the arm is not in the pre-approach zone. [LS-113]
	LS-112: The system uses a cached medication tag check result, assuming it is still valid. [UCA-20]	SR-102: Cached tag values must be invalidated unless verified at the correct time and position. [LS-112]
	LS-113: Incorrect arm position feedback causes the tag check to occur prematurely or be skipped. [UCA-20]	SR-103: Camera initialization and readiness must be verified by the pharmacy employee on the system screen, at least at the beginning of the workday. [LS-114, LS-118]
	LS-114: The tag check logic is enabled, but the camera is turned off or not initialized during execution. [UCA-20]	SR-104: If the camera is non-functional, misaligned, or obstructed, the system must halt and issue an alert. [LS-114, LS-115]
	LS-115: The camera is functional but misaligned, resulting in failure to read the tag correctly. [UCA-20]	SR-105: The robotic arm must not proceed to grasp or suction unless a tag is successfully recognized and matched. [LS-111, LS-119]
	LS-116: The lighting conditions prevent the camera from capturing the tag image with sufficient clarity. [UCA-20]	SR-106: The system must include checks for camera stream freshness, and reject verification based on outdated or frozen frames. [LS-117]
	LS-117: The camera feed is delayed or frozen, leading to validation on outdated visual information. [UCA-20]	SR-107: Lighting conditions in the storage area must comply with camera requirements to ensure tag readability. [LS-116]
	LS-118: The tag check routine does not include verification of camera status before executing. [UCA-20]	SR-108: Logs must include verification that the camera was active, the tag was captured, and the result matched the expected value. [LS-119]
	LS-119: The system logs a successful tag check despite camera failure or no tag detected. [UCA-20]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-21: Robotic Arm Controller performs "Check medication tag", but the tag is associated with the wrong item [H1]	LS-120: The robotic arm scans the tag correctly, but the tag was physically attached to the wrong medication. [UCA-21]	SR-109: The robotic controller must validate full tag content against the expected medication ID, batch, and type. [LS-120, LS-126, LS-127]
	LS-121: The tag data is correct, but the robot selects the wrong item from the shelf due to a misalignment. [UCA-21]	SR-110: Tag verification must include a real-time query to the central prescription or medication database. [LS-122, LS-124]
	LS-122: The tag matches a previous batch of medication but was not updated for the current prescription. [UCA-21]	SR-111: The system must reject tag matches if any ambiguity or duplication is found. [LS-126, LS-127]
	LS-123: A medication substitution occurred, but the database was not updated, so the robot accepts the incorrect tag. [UCA-21]	SR-112: If a tag mismatch is detected, the robot must halt the operation and notify the Support Team and Pharmacy Employee. [LS-120, LS-125]
	LS-124: The tag is associated with a different patient or prescription and not cross-validated at the time of pickup. [UCA-21]	SR-113: Interface notifications must highlight mismatch details (expected vs. read values) before action is allowed. [LS-125]
	LS-125: The interface does not notify the operator of a tag-mismatch, even though the system detected a discrepancy. [UCA-21]	SR-114: Medication storage must ensure correct physical placement and labeling of tags, with periodic verification. [LS-120, LS-121]
	LS-126: The controller compares tag IDs only partially (e.g., prefix match), resulting in false positives. [UCA-21]	SR-115: Every tag used in the dispensing system must be unique, traceable, and auditable. [LS-127]
	LS-127: The tag reading logic accepts duplicate IDs from different medications without error. [UCA-21]	SR-116: A substitution protocol must ensure that replacement medications receive updated, validated tags. [LS-123]
UCA-22: Robotic Arm Controller performs "Check medication tag" prematurely, resulting in the detection of the wrong medication [H1]	LS-128: The robotic arm performs the tag check while still moving or before reaching the exact medication location. [UCA-22]	SR-117: Tag verification must be enabled only after the robotic arm confirms arrival and stabilization at the exact pickup position. [LS-128, LS-131]
	LS-129: Mechanical latency or lag in the arm's encoder causes a premature signal to the tag reader. [UCA-22]	SR-118: The system must reject or repeat any tag read that occurred while the arm was in motion or unstabilized. [LS-128, LS-131]
	LS-130: The tag of a nearby or adjacent medication is scanned due to early activation. [UCA-22]	SR-119: Sensors or encoders must confirm physical arm stability before initiating tag reading. [LS-129, LS-131]
	LS-131: Position feedback is received with delay, and tag verification occurs while the arm is in transition. [UCA-22]	SR-120: The robot must validate that the detected tag corresponds to the spatial coordinates of the intended medication location. [LS-130]
		SR-121: If a mismatch is detected between expected and scanned tag, the robot must halt and request manual validation. [LS-130, LS-131]
		SR-122: Tag scan logs must include timestamp, position ID, and motion status of the arm at the moment of verification. [LS-128, LS-131]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-23: Robotic Arm Controller performs "Check medication tag" too late, causing the medication not to be detected [H8]	LS-132: The robotic arm initiates the tag check after starting the grasp or suction, skipping the correct verification moment. [UCA-23]	SR-123: Tag verification must occur before any grasping or suctioning procedure is executed. [LS-132, LS-134]
	LS-133: The robotic arm moves past the tag location without triggering the check at the intended time. [UCA-23]	SR-124: The robotic system must enforce a strict sequence: arm arrives → arm stabilizes → tag is verified → grasp begins. [LS-133, LS-136]
	LS-134: The grasping action begins and occludes the tag before it is read by the camera. [UCA-23]	SR-125: If tag verification does not occur within the predefined time window, the system shall attempt the action a specified number of times. Upon repeated failure, it shall abort the operation and issue an alert to the appropriate interface. [LS-132, LS-135]
	LS-135: Tag verification is delayed due to processing latency or overloaded resources. [UCA-23]	
	LS-136: A misconfiguration causes the robot to wait for excessive stabilization time before checking the tag. [UCA-23]	SR-126: The visual field required for tag detection must remain unobstructed until successful verification is confirmed. [LS-134, LS-137]
	LS-137: A delay in image processing results in tag recognition after the arm has already left the scan region. [UCA-23]	SR-127: Tag verification must be tightly bound to arm position feedback to prevent misaligned execution. [LS-133, LS-136]
	LS-138: Manual activation of tag check is not synchronized with the medication approach routine. [UCA-23]	SR-128: All tag check operations must generate a timestamped event with arm position and verification status for traceability. [LS-132, LS-138]
UCA-24: Robotic Arm Controller provides "Check medication tag" but ends before full read [H8]	LS-139: The tag check is interrupted by the start of a movement command, ending the scan prematurely. [UCA-24]	SR-129: The Robotic Arm Controller must not confirm tag verification unless all required tag fields have been read and matched. [LS-140, LS-145]
	LS-140: The scan completes only a portion of the tag (e.g., partial barcode or blurred QR code), but the system accepts it as valid. [UCA-24]	SR-130: Movement commands must be blocked while tag reading is in progress and until full verification is completed. [LS-139, LS-144]
	LS-141: The camera read times out before the full tag data is acquired. [UCA-24]	SR-131: The tag verification function must include validation steps for format completeness, clarity, and checksum (if applicable). [LS-140, LS-143]
	LS-142: The tag image is captured but not processed entirely due to system lag. [UCA-24]	SR-132: Timeouts during scanning must trigger an automatic retry or error notification instead of ending silently. [LS-141, LS-142]
	LS-143: The controller receives a tag verification success flag despite the tag data being incomplete. [UCA-24]	SR-133: Tag scan confirmation must log detailed content (fields acquired), time of completion, and image/frame integrity. [LS-144, LS-145]
	LS-144: The tag verification process is marked as complete due to a software logic bug when only the first segment of data is read. [UCA-24]	SR-134: Visual or auditory interface feedback must clearly indicate if tag reading was completed successfully or prematurely. [LS-139, LS-140]
	LS-145: No verification step checks whether the tag scan contains all required metadata. [UCA-24]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-25: Robotic Arm Controller does not provide "Control movements" while the robotic arm is in motion [H2, H5, H6, H7, H8]	LS-146: The Robotic Arm Controller sends an initial movement command but stops controlling it mid-motion. [UCA-25]	SR-135: Motion control must remain active and synchronized with sensor feedback throughout the full movement cycle. [LS-146, LS-148]
	LS-147: The robotic arm enters an unregulated state due to a software crash or control loop interruption. [UCA-25]	SR-136: Any interruption in motion control (software fault or module conflict) must halt the robot and raise an alert. [LS-147, LS-152]
	LS-148: Sensor feedback (e.g., encoder, joint position) is not processed during motion, leading to incorrect arm trajectory. [UCA-25]	SR-137: Movement commands must be continuously validated and adjusted according to real-time position, velocity, and orientation. [LS-146, LS-150]
	LS-149: The controller fails to apply braking or path corrections when obstacles or deviations are detected. [UCA-25]	SR-138: The system must apply safety corrections (e.g., slowdown, redirection) dynamically based on feedback. [LS-149, LS-153]
	LS-150: A communication delay causes motion adjustments to arrive too late to avoid impact or overshoot. [UCA-25]	SR-139: The Robotic Arm Controller must confirm motion completion only after receiving position convergence confirmation. [LS-151]
	LS-151: The control module assumes the trajectory is complete before motion actually ends. [UCA-25]	SR-140: All motion control operations must be logged with timestamps, status of control loop, and any applied corrections. [LS-146, LS-150]
	LS-152: A secondary process (e.g., vision system or interface update) disables control functions temporarily. [UCA-25]	SR-141: The system must enforce speed and torque constraints continuously during motion. [LS-153]
	LS-153: The robotic arm exceeds force or speed limits due to absence of continuous regulation during motion. [UCA-25]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-26: Robotic Arm Controller provides "Control movements" in incorrect sequence [H4, H6, H8]	LS-154: The robot repeats a previously executed movement (e.g., rotates twice), causing positional misalignment. [UCA-26]	SR-142: All movement sequences must be stored in validated templates and executed without modification unless explicitly authorized. [LS-157]
	LS-155: A movement step is skipped due to a sensor not acknowledging the previous step completion. [UCA-26]	SR-143: The Robotic Arm Controller must confirm that each movement has been completed and acknowledged before sending the next command. [LS-155, LS-156]
	LS-156: The arm retracts without confirming successful grasp, leading to loss of the medication. [UCA-26]	SR-144: The system must reject movement commands that would execute out of the expected logical order. [LS-157]
	LS-157: A system update or logic error reorders the default motion sequence. [UCA-26]	SR-145: Each motion command must include dependency checks to verify that the arm is in the correct state before proceeding. [LS-158]
	LS-158: Feedback latency causes the controller to move to the next step before the previous one is finalized. [UCA-26]	SR-146: Sensor and feedback delays must be handled with buffer or confirmation logic before motion continuation. [LS-155, LS-158]
		SR-147: Unexpected or repeated movement sequences must be logged and flagged for review. [LS-154, LS-157]
		SR-148: Manual overrides must be restricted during automated sequences unless a safety interlock is triggered. [LS-157]
UCA-27: Robotic Arm Controller does not provide "Grasp/suction medicine" during pickup [H8]	LS-159: The arm reaches the pickup position but does not activate the gripper or suction device. [UCA-27]	SR-149: The robotic arm must activate the grasp/suction only after confirming the arm is precisely aligned with the pickup target. [LS-159, LS-161]
	LS-160: A signal to activate the gripper/suction is lost due to a communication delay. [UCA-27]	SR-150: Gripper/suction activation must be logged and timestamped, including position and module status. [LS-160, LS-162]
	LS-161: The gripper/suction module is activated too early or too late, missing the medication's exact position. [UCA-27]	SR-151: All grasping actions must require confirmation from force/vacuum/contact sensors before continuing. [LS-163, LS-166]
	LS-162: A hardware malfunction prevents the suction or grasping mechanism from responding. [UCA-27]	SR-152: If no object is detected post-grasp/suction, the arm must stop and raise a "pickup failure" exception. [LS-164, LS-165]
	LS-163: The system assumes that the grasp/suction was successful without receiving confirmation from the sensor. [UCA-27]	SR-153: Feedback from the suction/gripper sensor must be continuously monitored and compared with expected pickup results. [LS-162, LS-163]
	LS-164: The medication is displaced or not in the expected position, and the grasping/suction action is ineffective. [UCA-27]	SR-154: The robotic system must retry or escalate when the grasp/suction attempt fails silently (e.g., suction fails but is not flagged). [LS-164, LS-165]
	LS-165: The suction module is clogged or partially obstructed, causing the grasp to silently fail. [UCA-27]	SR-155: System configurations that disable safety checks (e.g., for testing) must require explicit reactivation for operational use. [LS-166]
	LS-166: The system disables feedback confirmation due to performance settings or debugging mode. [UCA-27]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-28: Robotic Arm Controller provides "Grasp/suction medicine" in the wrong position [H3]	LS-167: The robot executes the grasp/suction command when the arm is off by a few centimeters from the target medication. [UCA-28]	SR-156: The robot must validate arm position using multiple sensors (e.g., encoders + camera) before initiating grasp. [LS-167, LS-169]
	LS-168: The medication has shifted on the shelf, but the robot uses a fixed coordinate without updating the position. [UCA-28]	SR-157: The robot must not perform a pickup if the object is not centered within the grasping/suction device tolerance zone. [LS-168, LS-172]
	LS-169: The system misinterprets feedback from the encoders, believing the arm is correctly positioned. [UCA-28]	SR-158: The robot must revalidate medication position immediately before executing grasp/suction. [LS-170, LS-171]
	LS-170: A tag was read correctly, but the robot aligned with a previous medication location. [UCA-28]	SR-159: Grasp/suction actions must be repeated if the arm is not fully stabilized at the computed target. [LS-174]
	LS-171: The camera detects the medication, but the grasp/suction is executed without confirming arm-tip alignment. [UCA-28]	SR-160: All medication grasp/suction locations must be dynamically calculated and confirmed via sensor fusion, not fixed coordinates alone. [LS-168, LS-173]
	LS-172: Minor misalignments (e.g., tilts or elevation errors) are ignored by the control logic due to tolerance settings. [UCA-28]	SR-161: Positioning discrepancies beyond threshold must trigger a repositioning attempt or escalation. [LS-172, LS-174]
	LS-173: The robot executes the pickup in the wrong quadrant due to incorrect transformation of coordinates. [UCA-28]	SR-162: All grasping operations must be logged with target vs. actual position data. [LS-167, LS-170]
	LS-174: The grasp/suction is activated while the robot is still in motion, causing inaccurate final positioning. [UCA-28]	
UCA-29: Robotic Arm Controller provides "Grasp/suction medicine" before positioning [H8]	LS-175: The grasp/suction command is triggered before the arm reaches the target due to missing position validation logic. [UCA-29]	SR-163: The Robotic Arm Controller must verify real-time arm position and stabilization before allowing grasp/suction commands. [LS-175, LS-177]
	LS-176: The robot begins suction during motion, assuming it is already aligned with the medication. [UCA-29]	SR-164: Grasping or suctioning actions must be blocked at the software level unless an explicit "position ready" signal is received. [LS-178]
	LS-177: A timing bug in the control algorithm causes the grasp/suction to occur slightly before arm stabilization. [UCA-29]	SR-165: During motion or transitional states, all grasp/suction routines must be inactive and disabled. [LS-176]
	LS-178: The control system lacks a blocking mechanism that ties grasping to arm position status. [UCA-29]	SR-166: The motion control and grasp control modules must be synchronized to ensure that suction/grasp occurs only after positioning is complete. [LS-175, LS-177]
		SR-167: The system must log the exact position, movement state, and confirmation status before every grasp/suction execution. [LS-175–LS-178]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-30: Robotic Arm Controller initiates "Grasp/suction medicine", but terminates the action before ensuring secure held [H4,H8].	LS-179: The grasp/suction command is deactivated based on a timer or arbitrary duration, not confirmation of secure grip. [UCA-30]	SR-168: The grasping/suction operation must remain active until secure-hold feedback is positively received and validated. [LS-179, LS-181]
	LS-180: The suction pressure or gripping force is not maintained long enough to secure the medication. [UCA-30]	SR-169: The system must monitor gripping force or suction pressure during the entire grasp phase and ensure it remains above threshold until completion. [LS-180, LS-183]
	LS-181: The sensor falsely reports completion of the grasp/suction device while the medication is only partially held. [UCA-30]	SR-170: The arm must not initiate retraction or further movement unless "secure held" status is confirmed. [LS-182]
	LS-182: The robotic arm begins retraction before verifying successful suction or grip, causing the item to be dropped. [UCA-30]	SR-171: Any grasping/suction operation that ends without secure confirmation must trigger an alert and halt the task. [LS-179, LS-182]
	LS-183: A glitch in the suction or gripper hardware interrupts the secure hold phase prematurely. [UCA-30]	SR-172: All grasp/suction completions must be logged with timestamp, hold verification result, and final state of medication. [LS-179, LS-183]
UCA-31: Robotic Arm Controller does not provide "Hold medicine" after grasping [H4]	LS-184: The controller releases suction or grip immediately after detecting initial contact, without maintaining hold. [UCA-31]	SR-173: After grasping, the system must maintain suction or grip force until a confirmed release is commanded. [LS-184, LS-185]
	LS-185: A software misconfiguration causes the hold routine to be skipped after the grasp/suction is complete. [UCA-31]	SR-174: The robot must enter a "hold state" explicitly after every successful grasp to ensure continuity of control. [LS-185, LS-187]
	LS-186: Gripper/suction pressure drops below minimum threshold before the robot reaches the delivery location. [UCA-31]	SR-175: Real-time grip/suction device status must be monitored and validated throughout the transport phase. [LS-186, LS-188]
	LS-187: The feedback sensor confirms grasp/suction, but the robot logic does not transition into the hold state. [UCA-31]	SR-176: Any grip pressure drop or suction loss during holding must trigger an alert and provide the option to restart the delivery of the current medicine. [LS-186, LS-188]
	LS-188: Vibrations or motion disrupt the hold, but no corrective action is taken because holding confirmation is not continuously checked. [UCA-31]	SR-177: All hold phases must be logged with start time, pressure/suction feedback, and release time. [LS-184–LS-189]
	LS-189: A task interruption (e.g., emergency stop or reassignment) causes hold logic to be aborted prematurely. [UCA-31]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-32: Robotic Arm Controller provides "Hold medicine" too tightly [H3]	LS-190: The robot applies default grip pressure unsuitable for fragile medication like ampoules. [UCA-32]	SR-178: The Robotic Arm Controller must define and enforce maximum pressure/suction thresholds per medication type. [LS-190, LS-192]
	LS-191: The suction pressure exceeds the limit necessary, causing deformation or damage to the medication. [UCA-32]	SR-179: Calibration routines must be periodically validated to ensure pressure limits remain accurate. [LS-194]
	LS-192: The system fails to differentiate between object types and applies the same grip configuration. [UCA-32]	SR-180: Object identification logic must inform grip configuration before the grasp or suction begins. [LS-192, LS-195]
	LS-193: The feedback sensor misreports the force applied, leading the controller to continue increasing pressure. [UCA-32]	SR-181: If sensor data suggests over-tightening or unsafe pressure buildup, the system must trigger an on-screen alert requesting corrective adjustments. [LS-191, LS-195]
	LS-194: The calibration values for pressure limits are incorrect or outdated, resulting in excessive holding force. [UCA-32]	SR-182: The system must log all grasp pressure values and object types during each holding operation for traceability. [LS-190, LS-193, LS-195]
	LS-195: The robot initiates a compensation routine assuming loss of grip and unintentionally over-tightens. [UCA-32]	
UCA-33: Robotic Arm Controller provides "Hold medicine" inappropriately [H4]	LS-196: The robot holds the medication at an unstable or curved point, compromising stability during transport. [UCA-33]	SR-183: The system must validate that the suction or grip point is physically stable and aligned with the medication's center of mass. [LS-196, LS-197]
	LS-197: The suction point is offset or misaligned, causing tilting or uneven pressure on the object. [UCA-33]	SR-184: Holding logic must avoid fragile, moving, or detachable parts of the object's surface. [LS-198]
	LS-198: The robot grasps a non-structural area of the package (e.g., label edge, cap), risking detachment. [UCA-33]	SR-185: Object geometry must be accounted for in selecting the suction/grip point. [LS-197, LS-200]
	LS-199: The medication is partially held or held at an angle, increasing risk of drop during arm movement. [UCA-33]	SR-186: The system must include orientation correction if the medication is not aligned with the holding vector. [LS-199]
	LS-200: A flat suction surface is assumed but the medication has an irregular or porous geometry. [UCA-33]	SR-187: All inappropriate holding attempts must be logged and flagged for future diagnostic analysis. [LS-196–LS-201]
	LS-201: The arm attempts to hold multiple items at once inappropriately, misbalancing grip force. [UCA-33]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-34: Robotic Arm Controller provides "Hold medicine" too late [H4]	LS-202: A buffering delay between grasp and hold routines causes the robot to lose grip. [UCA-34] LS-203: Suction pressure ramps up too slowly, allowing the item to shift or fall. [UCA-34] LS-204: A sensor misreads the grasp confirmation, postponing hold activation. [UCA-34] LS-205: Feedback from a previous task interferes with the timing of the hold routine in the current one. [UCA-34] LS-206: The robot begins moving before the "hold" phase has properly started. [UCA-34]	SR-188: Hold action must be executed immediately following successful grasp detection, with no procedural gaps. [LS-204] SR-189: The Robotic Arm Controller must maintain synchronization between grasp and hold subsystems. [LS-202, LS-205] SR-190: Any delay in hold activation beyond a configured threshold must trigger an exception and retry. [LS-203, LS-205] SR-191: The robot must not initiate arm movement until hold state is confirmed as active. [LS-206] SR-192: Gripper/suction ramp-up timing must be verified during validation and monitored during operation. [LS-203] SR-193: Task-level interference (e.g., leftover feedback) must be cleared before new grasp/hold cycles. [LS-205]
UCA-35: Robotic Arm Controller provides "Hold medicine" but ends before drop point [H4]	LS-207: The hold function is terminated prematurely due to a misinterpreted arrival at the drop point. [UCA-35] LS-208: An incorrect trajectory or speed causes the release before reaching the bag zone. [UCA-35] LS-209: System latency causes the grip to be released just before reaching the actual destination. [UCA-35] LS-210: Manual override or system misconfiguration disables the holding function too early. [UCA-35] LS-211: The robot releases the medication while adjusting its final angle or orientation before dropping. [UCA-35]	SR-194: The hold function must remain active until a precise drop-point confirmation is received. [LS-207, LS-209] SR-195: Any premature release of the hold state must trigger an emergency interrupt and log the event. [LS-209, LS-210] SR-196: The system must validate that grip or suction is maintained through the entire movement and delivery cycle. [LS-207, LS-208, LS-211] SR-197: All hold/release events must include drop-point coordinates, position accuracy, and grip status at release. [LS-207–LS-211]
UCA-36: Robotic Arm does not execute "Drop medicines" in medicine bag when the arm is correctly positioned [H8]	LS-212: The system confirms the correct position, but the drop command is never issued due to software omission. [UCA-36] LS-213: The robotic arm waits for redundant confirmation that never arrives, preventing the drop. [UCA-36] LS-214: A position sensor delay causes the controller to skip the drop sequence while already in position. [UCA-36] LS-215: The robot receives conflicting signals about drop readiness and cancels the drop. [UCA-36]	SR-198: The drop action must be triggered as soon as position confirmation and alignment are achieved. [LS-212, LS-213] SR-199: Position feedback must be verified from redundant sources or within a confirmation window to avoid unnecessary blocking. [LS-214, LS-215] SR-200: Drop command routines must include fault-tolerant logic to recover from minor signal delays. [LS-213] SR-201: The system must log position confirmation, time to release, and any blocked drop events for diagnostics. [LS-212]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-37: Robotic Arm executes "Drop medicines" when there is no bag available [H3, H8]	LS-216: The robotic arm drops the medication due to outdated or missing bag presence feedback. [UCA-37] LS-217: A false-positive from the medicine bag sensor causes the system to believe a bag is in place when it is not. [UCA-37] LS-218: The drop is triggered automatically before the bag reaches its correct position. [UCA-37] LS-219: The pharmacy employee forgets to place the bag before starting the process. [UCA-37] LS-220: The pharmacy employee incorrectly assumes that the bag is in place due to a visual obstruction or distraction. [UCA-37] LS-221: A race condition occurs: the bag is removed by someone just before the drop happens. [UCA-37] LS-222: The drop logic does not include a final bag presence verification before executing the release command. [UCA-37]	SR-202: Pharmacy employees must confirm visually and via the interface that the bag is present before initiating the robotic dispensing cycle. [LS-216, LS-217, LS-219, LS-220] SR-203: The Robotic Arm Controller must block any drop command if bag presence is not confirmed immediately before the action. [LS-221, LS-222] SR-204: A last-second bag validation step must be embedded into the drop routine, even if earlier confirmation occurred. [LS-218, LS-221] SR-205: The system must log all drop attempts, including whether bag presence was confirmed, sensor readings, and user confirmations. [LS-216–LS-222]
UCA-38: Robotic Arm executes "Drop medicines" when it is not positioned directly above the bag [H3, H8].	LS-223: The robot executes the drop command based on a position estimate that is off by a few centimeters. [UCA-38] LS-224: Position sensor drift causes the robot to misinterpret its current location. [UCA-38] LS-225: The system ignores minor misalignment due to wide tolerance parameters, leading to partial or failed delivery. [UCA-38] LS-226: A previous successful drop masks an unnoticed misalignment now repeated. [UCA-38] LS-227: The bag was moved slightly by external interference, and the arm was not re-centered before dropping. [UCA-38]	SR-206: The Robotic Arm Controller shall verify its current position against the Process Model prior to initiating the drop action. [LS-223, LS-224] SR-207: A positional accuracy check must be executed with defined tolerances based on the bag's geometry and location. [LS-225, LS-226] SR-208: After any movement of the bag or robot base, the system must re-validate alignment before drop. [LS-227] SR-209: The robot must reject any drop request if position error exceeds tolerance, and must raise an alert. [LS-223–LS-227] SR-210: All drops must be logged with arm coordinates, position error margin, and alignment confirmation. [LS-223–LS-227]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-39: Robotic Arm executes "Drop medicines" too late, when the bag has already been removed [H3, H8]	LS-228: There is a delay between position confirmation and drop execution, during which the bag is removed. [UCA-39]	SR-211: The system must recheck bag status presence within a short time window before drop execution. [LS-228, LS-230]
	LS-229: The robot ignores last-second bag removal due to reliance on earlier bag status state. [UCA-39]	SR-212: Any drop command must be canceled if the bag status is changed before or during drop execution. [LS-229, LS-232]
	LS-230: No continuous monitoring of bag presence is performed between positioning and drop. [UCA-39]	SR-213: In queued or delayed task modes, environment revalidation must occur before every critical action. [LS-231]
	LS-231: The system is in queue-based execution mode and executes a delayed drop without revalidating environment state. [UCA-39]	SR-214: A warning must be shown on the interface if a bag removal is detected after alignment but before drop. [LS-229, LS-232]
	LS-232: Human intervention removes the bag while the robot is temporarily idle or waiting. [UCA-39]	SR-215: All drop attempts must log last-known bag presence timestamp, position data, and result. [LS-228]
UCA-40: The robotic arm executed the "Drop Medicines" too late, delivering in a bag intended for another patient [H1c]	LS-233: The assigned medicine bag was replaced with another patient's bag before the drop, but the system was not aware of the change. [UCA-40]	SR-216: The Robotic Arm Controller must confirm that the bag in place matches the intended patient for that medicine before drop. [LS-233, LS-234]
	LS-234: The system uses static bag-patient association without real-time validation at drop time. [UCA-40]	SR-217: Any mismatch between ticket and bag ID at the time of drop must block the action and raise an alert. [LS-235, LS-236]
	LS-235: No cross-verification occurs between the medicine ticket and the currently positioned bag. [UCA-40]	SR-218: Bag-patient verification must be performed in real-time just before medicine release. [LS-234]
	LS-236: A rushed manual replacement during robot movement leads to unintended bag reassignment. [UCA-40]	SR-219: If a bag replacement is detected after robot alignment, the system must force reassignment or cancellation of the drop. [LS-233, LS-236]
		SR-220: All patient-bag associations must be logged with timestamps, medicine ID, and final drop confirmation status. [LS-233–LS-236]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-41: Robotic Arm does not "Get medicines" when a dispense command has been issued [H8]	LS-237: The dispense command is issued, but the robot fails to initiate action due to internal queue delays. [UCA-41]	SR-221: The Robotic Arm Controller must ensure that every valid dispense command results in the initiation of retrieval, in accordance with a predefined dispensing list. [LS-237, LS-241]
	LS-238: A task conflict or robot resource lock prevents the command from being executed. [UCA-41]	SR-222: Robotic task queues must prioritize active dispense commands over non-critical tasks. [LS-237, LS-238]
	LS-239: The dispense command is received but not parsed correctly due to formatting or data errors. [UCA-41]	SR-223: All dispense commands must undergo validation and error-checked parsing before triggering execution. [LS-239]
	LS-240: The dispense instruction is lost in communication between the interface and the robotic controller. [UCA-41]	SR-224: Communication between systems must include delivery acknowledgment and execution confirmation for all dispense commands. [LS-240, LS-241]
	LS-241: The system logs the dispense command, but the execution trigger to the arm never activates due to a logic bug. [UCA-41]	SR-225: If a dispense command fails to trigger robotic movement, an alert must be issued and retry logic initiated. [LS-238, LS-240]
	LS-242: The robotic arm is powered off, and the pharmacy employee is unaware of its offline state. [UCA-41]	SR-226: The system must continuously monitor robot status (power, fault state, position) and prevent command loss when the robot is unavailable. [LS-242, LS-243, LS-245]
	LS-243: The robotic arm is in a failure state and cannot trigger the dispense routine. [UCA-41]	
	LS-244: A physical obstacle is blocking the robot's movement, preventing it from initiating the pickup. [UCA-41]	SR-227: Pharmacy employees must be notified immediately if the robot is offline or in a state that blocks dispensing. [LS-242, LS-243]
	LS-245: The robot is mechanically stuck in a previous position that restricts it from initiating the medicine retrieval sequence. [UCA-41]	SR-228: The robotic controller must detect physical obstructions and halt operation with an appropriate error code. [LS-244]
		SR-229: Recovery protocols and training must exist to reposition the robot if it becomes stuck after a prior routine. [LS-245]
UCA-42: Robotic Arm "Get medicines" without a corresponding valid command [H1, H3, H4, H8]	LS-246: A leftover dispense command from a previous session is re-triggered without revalidation. [UCA-42]	SR-230: The Robotic Arm Controller must validate the integrity and origin of every dispense command before initiating "Get medicines." [LS-246]
	LS-247: A duplicated or erroneously repeated dispense command causes the robot to fetch the same medicine twice. [UCA-42]	SR-231: Execution of dispense commands must be linked to active session identifiers and logged to prevent reuse. [LS-246, LS-247]
	LS-248: The robot interprets internal test routines or diagnostics as real dispense commands. [UCA-42]	SR-232: All commands must be confirmed as non-canceled and currently assigned before robot movement is allowed. [LS-249]
	LS-249: An operator accidentally issues a dispense command for a canceled or reassigned ticket. [UCA-42]	SR-233: Test, diagnostic, or maintenance routines must be segregated from operational dispense logic. [LS-248]
	LS-250: The robotic system triggers a medicine pickup due to a misfire in the interface logic or synchronization error. [UCA-42]	SR-234: The system must alert the operator if a medicine retrieval is triggered without a corresponding valid request. [LS-250]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-43: Robotic Arm provides "Get medicines" from incorrect shelf [H1, H3, H4]	LS-251: The robotic controller receives an outdated or incorrect shelf ID due to delayed database synchronization. [UCA-43]	SR-235: Before initiating pickup, the Robotic Arm Controller must validate that the shelf ID matches the one linked to the current medicine and dispense ticket. [LS-251, LS-252]
	LS-252: A dispense command is linked to the wrong shelf location due to incorrect medication–shelf mapping. [UCA-43]	SR-236: The shelf ID must be confirmed against the latest centralized mapping and reflect the current dispense session. [LS-251, LS-253]
	LS-253: The shelf ID in the database was altered without proper verification or audit logging. [UCA-43]	SR-237: Shelves with ambiguous or unreadable tags must trigger a halt in operation and raise an alert. [LS-254, LS-255]
	LS-254: The robot misinterprets a QR code or visual label and moves toward the wrong shelf. [UCA-43]	SR-238: The robotic navigation system must include constraints that prevent deviation to unverified shelves, even if nearby. [LS-256]
	LS-255: Two shelf IDs are similar in format or physical layout, causing ambiguity in shelf recognition. [UCA-43]	SR-239: A complete log of shelf selection, shelf ID, and corresponding ticket must be recorded for every dispense cycle. [LS-251–LS-256]
	LS-256: The robot reroutes to a physically closer shelf when facing minor navigation error, despite incorrect ID. [UCA-43]	
UCA-44: Robotic Arm provides "Get medicines" of the wrong type (blister/ampoule) [H3, H4, H8]	LS-257: The medication type in the database is outdated or mismatched with the actual physical inventory. [UCA-44]	SR-240: The system must validate that the medication type assigned in the dispense ticket matches the physical type detected before initiating pickup. [LS-257, LS-261]
	LS-258: The Robotic Arm Controller misinterprets the item tag or visual marker, leading to the wrong type being fetched. [UCA-44]	SR-241: Visual or code-based identification must confirm the medication's physical type in addition to identity. [LS-258, LS-262]
	LS-259: The gripper or suction device is configured for one type (e.g., blister) but the medication to be picked up is another type (e.g., ampoule). [UCA-44]	SR-242: The robotic gripper/suction system must be configured and validated to match the required medication type before execution. [LS-259]
	LS-260: The system fails to validate the physical type of the medication before initiating the pickup action. [UCA-44]	SR-243: If the physical type detected does not match the expected type, the system must abort the operation and alert the operator. [LS-260]
	LS-261: A human operator inputs the wrong medication type in the dispense ticket. [UCA-44]	SR-244: All pickups must be logged with expected type, detected type, gripper/suction configuration, and confirmation status. [LS-257–LS-262]
	LS-262: The robotic arm fetches a similar-looking item without type-specific validation. [UCA-44]	

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-45: Robotic Arm provides "Get medicines" before item is available [H8]	LS-263: The robotic arm attempts to pick up the medication before it has been restocked on the shelf. [UCA-45]	SR-245: The robotic system must verify shelf status in real time using validated inputs before initiating a pickup. [LS-263]
	LS-264: A synchronization error causes the system to believe the shelf has been updated when it has not. [UCA-45]	SR-246: Shelf and inventory systems must be synchronized to reflect true item availability status. [LS-264, LS-265]
	LS-265: The item is in transit from storage to shelf but the command is prematurely issued. [UCA-45]	SR-247: If item presence is not confirmed within a defined window, the system must block the pickup and raise an alert. [LS-266]
	LS-266: The operator confirms shelf readiness in the system, but the item is not physically present yet. [UCA-45]	SR-248: Manual confirmations by operators must be verified against shelf status input before enabling robotic actions. [LS-266]
		SR-249: Pickup actions must be logged with timestamp, shelf ID, item ID, and presence confirmation source. [LS-263–LS-266]
UCA-46: Robotic Arm "Get medicines" too early, before the correct medication ID or location is confirmed [H1, H3, H4]	LS-267: The robotic arm initiates pickup before receiving tag verification from the visual system. [UCA-46]	SR-250: The Robotic Arm Controller must confirm medicine ID and location via real-time tag verification before initiating any pickup. [LS-267, LS-268]
	LS-268: The medicine ID is not confirmed because the shelf tag is obscured or damaged, but the pickup proceeds. [UCA-46]	SR-251: Retrieval actions must be blocked if either the tag or position data is missing, outdated, or inconsistent. [LS-269, LS-271]
	LS-269: The system uses an outdated medicine location map and initiates pickup without reconfirming the current position. [UCA-46]	SR-252: Visual confirmation must be required and compared to the expected data; discrepancies must raise alerts. [LS-270]
	LS-270: A mismatch occurs between the expected tag and the detected one, but the system does not halt the retrieval. [UCA-46]	SR-253: An alert must be generated and displayed on screen when tag recognition fails a predefined number of times, prompting secondary confirmation or manual correction. [LS-268, LS-271]
	LS-271: Tag recognition is skipped due to time-saving optimizations or sensor misconfiguration. [UCA-46]	SR-254: All retrieval operations must log the detected tag, position coordinates, confirmation method, and system decision trace. [LS-267–LS-271]
UCA-47: Robotic Arm provides "Get medicines" but the action is stopped too soon, preventing medication retrieval [H8]	LS-272: A timeout in the control logic causes the retrieval operation to end before the item is secured. [UCA-47]	SR-255: The Robotic Arm Controller must verify medication presence via sensor feedback before completing the retrieval sequence. [LS-272, LS-273]
	LS-273: Suction/gripper feedback is not properly interpreted, and the system assumes the action is complete. [UCA-47]	SR-256: If the retrieval action ends prematurely, the system must automatically trigger a retry or notify the operator. [LS-274, LS-275]
	LS-274: An operator manually interrupts the retrieval process without verifying that the medication has been collected. [UCA-47]	SR-257: The control algorithm must prevent task completion without confirmation of secure pickup. [LS-272, LS-275]
	LS-275: An internal exception or safety check forces a halt before pickup is complete, but no retry logic is executed. [UCA-47]	SR-258: The system must detect and react to instability or grip/suction loss during motion. [LS-276]
	LS-276: Vibration or misalignment causes early disengagement of the gripper or suction system. [UCA-47]	SR-259: All aborted or incomplete retrievals must be logged with reason, duration, and system state at interruption. [LS-272–LS-276]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-48: Support Team does not provide "Authorize operation" when robot is ready [H8]	LS-277: The Support Team is unaware that the robot has reached a ready state due to missed notification or alert. [UCA-48]	SR-260: The robot system must notify the Support Team when it enters a confirmed ready state, with visual and audible alerts. [LS-277, LS-278]
	LS-278: Interface or dashboard indicators fail to update correctly, showing the robot as unavailable. [UCA-48]	SR-261: Interface elements must reflect the actual robot state in real time and prevent misinformation. [LS-278]
	LS-279: The robot is ready, but the authorization interface is temporarily down or inaccessible. [UCA-48]	SR-262: The authorization tool must remain continuously accessible to the Support Team when robot status is "ready." [LS-279]
	LS-280: The Support Team postpones authorization awaiting unnecessary manual checks despite readiness. [UCA-48]	SR-263: Delays in authorizing a ready robot must trigger automatic alerts or escalation protocols. [LS-280, LS-281]
	LS-281: The robot is reinitialized and stabilized, but no automated signal prompts the team for authorization. [UCA-48]	SR-264: All authorization-related events must be logged with timestamp, operator ID, and robot state snapshot. [LS-277–LS-281]
UCA-49: Support Team provides "Authorize operation" when robot is unsafe [H6]	LS-282: The Support Team mistakenly authorizes operation due to misinterpreting system logs or warning messages. [UCA-49]	SR-265: The Support Team must verify that no unresolved safety faults are active before issuing authorization. [LS-282, LS-284]
	LS-283: A critical failure status is not displayed in the interface due to a rendering or refresh issue. [UCA-49]	SR-266: The robot interface must display comprehensive safety diagnostics, updated in real time. [LS-283, LS-285]
	LS-284: The Support Team issues authorization while a background process still reports instability. [UCA-49]	SR-267: Authorization requests must be blocked if background fault checks are still running. [LS-284]
	LS-285: Residual fault data is cleared by a system reset, giving the false impression of readiness. [UCA-49]	SR-268: System logs must retain fault traces even after resets, marking them as unresolved until manually reviewed. [LS-285]
	LS-286: Pressure from operations staff leads to rushed or incomplete safety verification. [UCA-49]	SR-269: All authorizations must be accompanied by a safety checklist and require explicit confirmation by the Support Team. [LS-286]
UCA-50: Support Team does not provide "Give support" when the pharmacy employee does not know what actions to take [H1-H8]	LS-287: The pharmacy employee signals a request for help, but the Support Team does not receive the notification due to communication failure. [UCA-50]	SR-270: The system must generate a clear and persistent support request notification when the pharmacy employee signals the need for assistance. [LS-287, LS-290]
	LS-288: The system lacks a clear "request support" interface or it is unintuitive, leading to missed requests. [UCA-50]	SR-271: The pharmacy employee interface must provide an intuitive and accessible option to request help. [LS-288]
	LS-289: The pharmacy employee becomes inactive in the interface due to confusion, but no inactivity alert is generated. [UCA-50]	SR-272: The system must monitor for inactivity or hesitation patterns and recommend Support Team intervention when appropriate. [LS-289]
	LS-290: The system logs the need for support, but no automatic routing to the responsible team is performed. [UCA-50]	SR-273: All support requests must be timestamped, logged, and escalated if not acknowledged within a specified timeframe. [LS-287, LS-291]
	LS-291: The Support Team overlooks or deprioritizes the support request because it was not marked as urgent. [UCA-50]	SR-274: The Support Team must be trained to recognize and prioritize situations of uncertainty reported by pharmacy employees. [LS-291]

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-51: Support team provide "Give support" with erroneous information [H1-H8]	LS-292: The Support Team uses outdated documentation containing incorrect procedures. [UCA-51]	SR-275: All support documentation must be validated, version-controlled, and regularly updated with traceable revision history. [LS-292, LS-293]
	LS-293: The documentation was updated recently, but the team was not informed of the changes. [UCA-51]	SR-276: A notification system must alert the Support Team when manuals or procedures are changed or superseded. [LS-293]
	LS-294: The Support Team misinterprets a procedure due to ambiguous or incomplete guidance. [UCA-51]	SR-277: Support Team members must be trained to recognize ambiguous instructions and seek confirmation before advising. [LS-294]
	LS-295: There is no formal procedure for the specific situation, and the team provides improvised or incorrect advice. [UCA-51]	SR-278: When guidance outside official documentation is required, it must be based on pre-approved fallback procedures. [LS-295]
	LS-296: A discrepancy exists between the behavior of the robot and what is described in the manual, leading to misguidance. [UCA-51]	SR-279: System behavior must be regularly compared against procedural documentation to ensure consistency. [LS-296]
UCA-52: Support Team provides "Give support" too late, after the error happen for long time[H1-H8]	LS-297: The system detects an error, but the alert to the Support Team is delayed due to communication lag or failure. [UCA-52]	SR-280: The system must immediately alert the Support Team when a process failure or user inactivity exceeds a critical threshold. [LS-297, LS-299]
	LS-298: The Support Team is not actively monitoring system status or user activity and misses signs of prolonged error. [UCA-52]	SR-281: All anomalies or errors must be prioritized in the interface and linked to user and robot logs for immediate context. [LS-300]
	LS-299: The pharmacy employee hesitates for a long time during the process, but no automatic alert is generated. [UCA-52]	SR-282: The Support Team must have access to real-time monitoring dashboards to detect lack of user action or unresolved errors. [LS-298]
	LS-300: An error message is logged but not prioritized or displayed prominently to the Support Team. [UCA-52]	SR-283: If no support action is registered within a defined timeframe after a fault, the system must trigger escalation. [LS-301]
	LS-301: The support escalation protocol is misconfigured or disabled, delaying the transfer of responsibility. [UCA-52]	

Source: Author (2025).

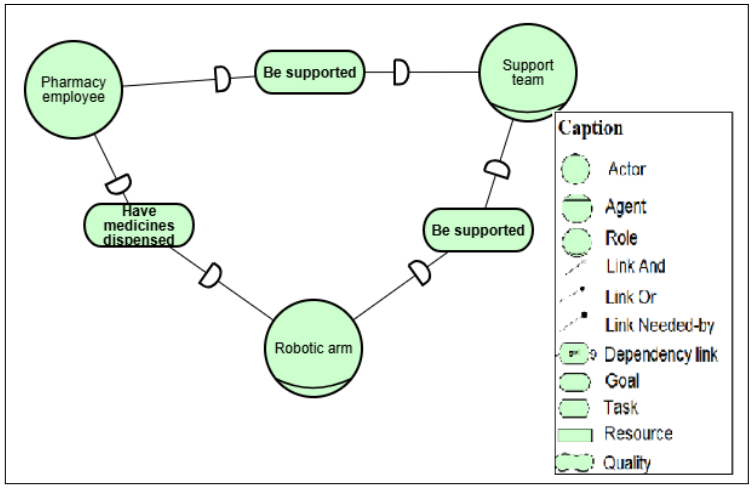
Important Notes: UCA-03 – *Pharmacy Employee provides "Check ticket-patient" too late in the dispensing process* [H1] did not have any loss scenarios identified. This is because the implementation of Safety Requirement **SR-01** – *The user interface must always display the mandatory "Check ticket-patient" option when the ticket status is "Waiting"* [LS-01, LS-04, LS-05, LS-06] ensures its mitigation through a software lockout mechanism. Therefore, it was a modeling decision not to define loss scenarios for this UCA.

4.6.6 Step 6 – Update the iStar4Safety Models

1. SD Model: This figure 4.9 presents the final SD model from Iteration 1 of the real Robotic Arm system for medication delivery in a hospital environment. Notably, no

changes were made to the initial SD model. This is mainly because no non-safety elements were introduced during this iteration, and iStar4Safety elements are not allowed to be of the dependency type.

Figure 4.9: Final SD Model of Iteration 1



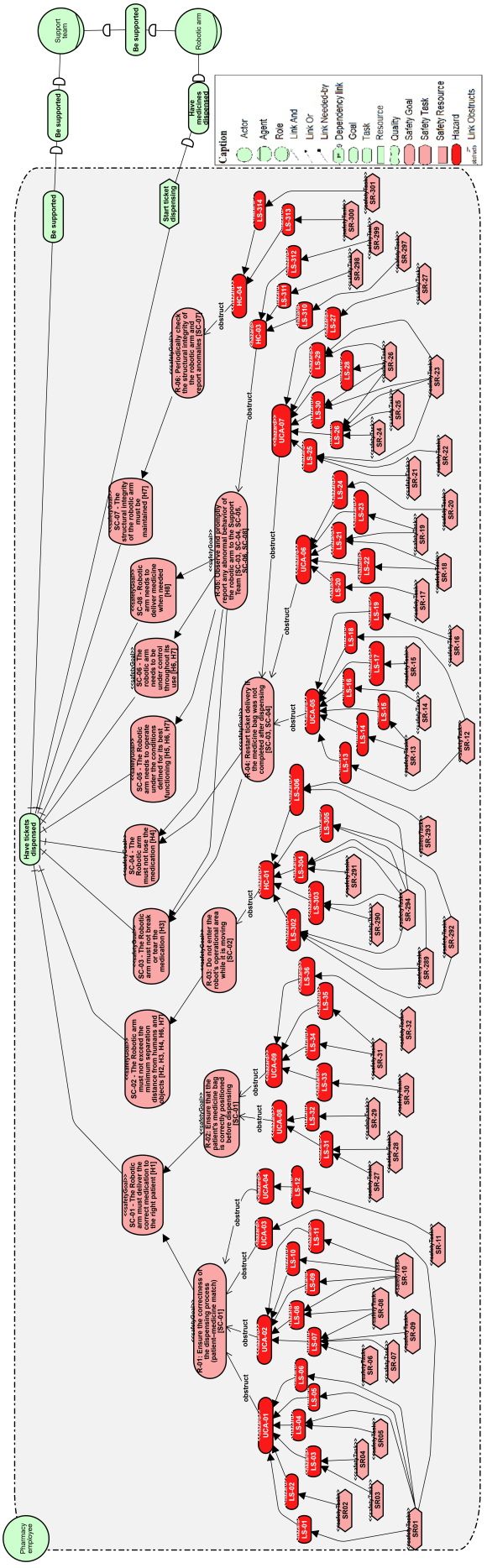
Source: Author (2025).

2. **SR Model:** Finally, due to space constraints, Figure 4.10 presents a partial SR model, in which the actor *Pharmacy Employee* is expanded, while the other actors—*Support Team* and *Robotic Arm*—remain collapsed. The complete SR model and its corresponding analysis are provided in Appendix F. The reader is referred to that section for the final models from this iteration of the RESafety process applied to the Robotic Medicine Delivery System, together with a detailed description of each actor individually.

The document generated as an artifact from the analysis of this illustrative scenario—based on a real robotic arm system for medication delivery in a hospital context—can be accessed at: ***Access the RESafety analysis document***.

4.7 Discussion

The application of the RESafety process to the Robotic Medicine Delivery System demonstrated its ability to integrate requirements engineering and safety engineering practices in a real-world context. By iteratively combining iStar4Safety and STPA, it was possible to model both organizational responsibilities and technical safety concerns in a systematic manner. The process supported the identification of losses, hazards, unsafe control actions (UCAs), hazardous conditions (HCs), safety constraints, and loss scenarios, all of which were refined into safety requirements within the scope of this first iteration of the process.



Source: Author (2025).

Figure 4.10: SR Model from Step 6 – Hybrid SR model with the Pharmacy Employee actor expanded

One of the strengths of the approach lies in its well-defined activities, which feed into subsequent phases and align stakeholder responsibilities with safety constraints. The use of iStar4Safety facilitated the visualization of dependencies among actors and components—such as the Pharmacy Employee, Robotic Arm, and Support Team—providing traceability from high-level goals to specific safety requirements. At the same time, the adoption of STPA made it possible to identify hazards emerging from system interactions rather than isolated component failures, which is particularly relevant in environments where humans and robots coexist.

Nonetheless, some limitations emerged. The first iteration produced a large number of elements—52 UCAs, 7 HCs, 59 safety constraints, 301 loss scenarios, and 283 safety requirements—which highlights the scalability challenge of applying the process to complex systems. This volume of artifacts also made the iStar4Safety models increasingly complex to represent, even though this was only the first iteration. While such richness of output supports a thorough analysis, it also raises concerns regarding analyst workload and the practicality of maintaining consistency across iterations. Furthermore, the focus on the robotic arm subsystem meant that broader organizational or hospital-level hazards were abstracted as environmental factors, leaving opportunities for future work to extend the analysis toward a system-of-systems perspective.

Finally, the case study confirmed that RESafety can contribute to bridging the gap between requirements and safety perspectives, while remaining adaptable to iterative refinement. Future research should focus on providing tool support to manage the large volume of generated artifacts, as well as on applying the process in other safety-critical domains to evaluate its generalizability.

4.8 Chapter Summary

This chapter presents the modeling of a safety-critical system—a robotic arm responsible for medicine delivery in a hospital pharmacy—studied over a six-month period.

Each step of the RESafety process is illustrated as applied to this system, with the corresponding artifacts generated and described in detail. As this represents the first iteration, additional details may be incorporated as the project evolves. In this iteration, a total of five losses, eight system-level hazards, eight safety constraints, and 21 responsibilities were identified in Step 1. In Step 4, 52 UCAs, seven HCs, and 59 controller constraints were generated. Step 6 identified 301 loss scenarios and 283 safety requirements aimed at mitigating them.

Moreover, the model itself may be refined in subsequent iterations based on the analysis of redundancies or trade-offs identified during the process.

The next chapter presents the expert-based evaluation of the RESafety process, involving specialists in safety and requirements engineering.

5

Expert Evaluation Through a TAM-Based Survey Instrument

This chapter presents the evaluation of the RESafety process through an empirical study, conducted with professionals who have experience in requirements engineering, safety engineering, or both. The assessment was carried out through a structured survey, and the following sections provide a detailed description of its design, execution, and analysis.

5.1 Introduction

According to Wohlin *et al.* (2012), conducting an empirical study is essential for evaluating processes and activities performed by humans. The survey method is an empirical research approach used to collect information to describe, compare, or explain knowledge, attitudes, and behaviors (Kitchenham & Pfleeger, 2008; Gorschek *et al.*, 2006). For this evaluation, a previous version of the RESafety process — which comprised seven steps — was employed. At that time, this was the structure of the process, which was later refined into the final version presented in this work. In that earlier version, Step 4 ended at the decision point where the analyst had to decide whether to proceed to substep "4.8 – Model HCs for responsibilities without UCAs, or add more HCs as needed" or to finish the step. Step 5 encompassed the activities related to formulating safety constraints. In the final version, as presented in Chapter 3 (Process Description), these two steps were merged, making the formulation of constraints part of Step 4. This change streamlined the workflow and reinforced the traceability between hazards, control actions, and safety constraints.

5.1.1 Justification for Using TAM

The Technology Acceptance Model (TAM), originally proposed by Davis Davis (1989), was adopted in this evaluation due to its strong theoretical foundation and extensive empirical validation in the field of technology adoption. TAM was specifically developed to explain user

behavior in relation to information technology (IT), centering on two core constructs: **Perceived Usefulness** (PU) and **Perceived Ease of Use** (PEOU).

In this thesis, TAM is applied to evaluate the RESafety process by providing a structured and reliable framework to understand participants' perceptions regarding its adoption and usability. The focus on usefulness and ease of use is closely aligned with our research objectives, as it allows us to examine whether the RESafety process is perceived as both valuable and user-friendly in the context of conducting safety analyses for safety-critical systems.

Through the lens of TAM, this study aims to generate meaningful insights into the acceptance and practical viability of the RESafety process, particularly in academic and professional environments involving requirements engineering and system safety.

5.1.2 Survey Steps

The main steps to be followed when conducting an empirical evaluation are (Kitchenham & Pfleeger, 2008):

- Define the objectives.
- Design the survey.
- Develop the survey instruments.
- Validate the survey instruments.
- Collect valid data.
- Analyze the collected data.

In the following sections, we provide a detailed description of the evaluation process, including the context in which it was conducted, the rationale for selecting the survey method, and the specific steps followed during its execution: defining the objectives, designing and validating the survey instruments, collecting valid data, and analyzing the responses. Finally, we present the results of the evaluation, discuss the key findings, and outline the conclusions drawn from this study. Additional methodological details are provided in Section 2.4.

5.1.3 Evaluation Context

The evaluation was implemented in the form of a survey (Wohlin *et al.*, 2012), using a self-administered questionnaire (Kitchenham & Pfleeger, 2008). Participants were selected through convenience sampling, a non-probability sampling technique in which individuals who are most easily accessible are chosen. The survey was directed at professionals representative of the intended users of RESafety, specifically requirements engineers and safety analysts. This survey is exploratory in nature, serving as a preliminary study to support a more comprehensive

investigation and to help ensure that no relevant issues are overlooked during the evaluation process.

Through the analysis of the evaluation results, our objective was to implement improvements and gain a deeper understanding of what experts expect from a process within the proposed context.

5.2 Expert Survey Methodology

5.2.1 Evaluation Objectives

Analyze the RESafety process
For the purpose of evaluation
With respect to perceived usefulness and perceived ease of use
From the point of view of requirements engineers and safety engineers
In the context of conducting safety analysis during requirements engineering.

To the best of our knowledge, the following research questions represent the inquiries we intend to address through the execution of our empirical study.

5.2.1.1 Research Questions

To guide our evaluation study, we established three research questions that this assessment of the RESafety process seeks to address:

- **Survey - RQ1 – How useful is the RESafety process for supporting the development of the safety analysis?** - This question seeks to assess the *Perceived Usefulness* of the RESafety process as perceived by requirements engineers and safety engineers. It is addressed through six survey items based on the Technology Acceptance Model (TAM).

- **Survey – RQ2 – How easy is it to use the RESafety process in the development of the safety analysis?** This question seeks to assess the *Perceived Ease of Use* of the RESafety process as perceived by requirements engineers and safety engineers. It is addressed through six survey items based on the Technology Acceptance Model (TAM).

- **Survey – RQ3 – How could the RESafety process be improved?** This research question seeks to gather feedback on the RESafety process from the perspective of requirements engineers and safety engineers, with a focus on identifying potential improvements. It is addressed through an open-ended question in the survey: *"Do you have any additional remarks? Please write them below. Your opinion is very*

important to our research." The goal is to collect suggestions and criticisms that may help enhance the RESafety process.

5.2.2 Expert Definition and Eligibility Criteria

To avoid ambiguity in participant selection andand to seek to ensure that the survey reflects informed professional judgment, explicit eligibility requirements were established to characterize who qualifies as an expert in this study. The term "expert" is operationalized through verifiable education, experience, and contribution indicators. Two expertise tracks were considered: Requirements Engineering (RE) and Safety Engineering (SE).

A participant was classified as an expert in RE or SE depending on their self-declared primary professional or research area, as indicated in the questionnaire item: *"What is your primary professional or research area in the field of safety and/or requirements engineering?"*. Participants were selected based on evidence of research publications and/or professional projects involving Requirements Engineering and/or Safety Requirements. More specifically, experts were chosen among researchers with recognized experience or academic work related to STPA and/or iStar (including iStar4Safety).

The following baseline conditions were required for inclusion:

- (B1) Minimum of three years of professional or research experience in Requirements Engineering or Safety Engineering.
- (B2) Basic familiarity with at least one of the techniques assessed in this study (iStar/iStar4Safety/GORE modeling and/or STPA), as verified through the participant's CV or public academic profile.
- (B3) Possession of a postgraduate degree (M.Sc., Ph.D., or Postdoctoral level).

Participants were excluded if they met any of the following conditions: (i) less than three years of professional or research experience; or (ii) inability to provide minimal verification evidence (e.g., institutional webpage, DBLP/Google Scholar, CV, LinkedIn, or project portfolio).

The established criteria aim to balance academic and industrial indicators in order to capture expertise relevant to early safety analysis and goal-oriented modeling. This operationalization reduces selection ambiguity, supports reproducibility, and aligns with established methodological recommendations for survey-based evaluations in software and systems engineering (Kitchenham & Pfleeger, 2008; Wohlin *et al.*, 2012).

5.2.3 Survey Design

The evaluation of the RESafety process consisted of tutorial videos and a questionnaire designed to collect participants' opinions, based on the Technology Acceptance Model (TAM).

The evaluation was conducted asynchronously to allow participants to engage with the materials at their convenience, ensuring they could complete the activity without time pressure or interference with their regular professional and personal duties.

Initially, we selected requirements engineers and safety engineers with expertise in STPA and invited them via email to voluntarily participate in our study.

For those who accepted the invitation, a follow-up email was sent containing detailed instructions for completing the evaluation. The email included:

- A link to access the evaluation materials.
- An estimate of the time required and a note emphasizing the asynchronous nature of the evaluation.
- Step-by-step instructions: watch the tutorial videos and complete the questionnaire.

After sending the evaluation access email, we allowed a total period of two weeks and two days for data collection.

5.2.4 Survey Instruments

Developing survey instruments involves searching the relevant literature, constructing the instrument, validating it, and documenting it appropriately (Kitchenham & Pfleeger, 2008).

In conducting our evaluation, we first reviewed the relevant literature to identify essential elements in the design of empirical surveys (Kitchenham & Pfleeger, 2008; Wohlin *et al.*, 2012). This review guided the definition of the key variables to be assessed and informed the structure of our evaluation strategy. The evaluation comprised two main components: (i) a set of tutorial videos designed to introduce and train participants in the RESafety process, and (ii) a self-administered online questionnaire grounded in the Technology Acceptance Model (TAM).

Questionnaires are the most widely used instrument in TAM-based research, followed by interviews (Börstler *et al.*, 2024). Building upon this, we developed a TAM-based questionnaire specifically targeted at Requirements Engineering and STPA professionals, aiming to gather their perceptions and assessments regarding the artifacts generated through the application of the RESafety process.

The TAM form consisted of twelve questions and is presented in Table 5.1. Each research question is linked to its respective construct, corresponding code, and the scale item used for evaluation. The questionnaire uses a seven-point Likert scale to measure participants' agreement with each statement. On this scale, 1 represents "Strongly disagree" and 7 represents "Strongly agree" with intermediate values indicating varying degrees of agreement or disagreement.

Table 5.1: Overview of TAM Questionnaire Items Adapted for RESafety

Construct	Code	Scale Item	Question
1. Perceived Usefulness (PU)	Q01	Work More Quickly	RESafety enables me to accomplish safety analysis tasks more quickly.
	Q02	Job Performance	Using RESafety improves my safety analysis.
	Q03	Increase Productivity	Using RESafety increases my productivity.
	Q04	Effectiveness	Using RESafety enhances my effectiveness in performing safety analysis.
	Q05	Makes Job Easier	Using RESafety makes it easier to do my safety analysis.
	Q06	Useful	Overall, I find RESafety useful for safety analysis of Safety Critical Systems.
2. Perceived Ease of Use (PEOU)	Q07	Easy to Learn	The use of RESafety does not require a lot of my mental effort.
	Q08	Controllable	I find it easy to follow the steps of RESafety.
	Q09	Clear & Understandable	The steps of RESafety are clear and understandable.
	Q10	Flexible	The steps of RESafety are flexible.
	Q11	Easy to Become Skillful	It is easy to become skillful in using RESafety.
	Q12	Easy to Use	I find the RESafety process easy to use.

Source: Author (2025).

In addition, an open-ended question was employed to elicit deeper insights that participants might have wished to provide, as open-ended questions are expected to capture information that closed-ended questions may not be able to uncover (Popping, 2015). The question was: “Do you have any additional remarks? Please write them below. Your opinion is very important to

our research." It was not mandatory to answer this question.

To validate the survey instruments, we conducted a pilot evaluation with postgraduate students in Computer Science. This preliminary round provided valuable insights that led to refinements and improvements to the evaluation materials.

Subsequently, we developed an online platform to host the evaluation, which included all necessary artifacts for assessing RESafety within the context defined for this research.

Finally, the instruments were the following:

- An online page containing tutorial videos, the questionnaire, and supporting materials. See Appendix A.
- A playlist of tutorial videos designed to provide participants with a basic understanding of the introductory concepts and the RESafety process:
    - Video 1: Introduction and Technology Overview – 7min20s. Link: `https://www.youtube.com/watch?v=7o4Rptpth8c`
    - Video 2: RESafety Process and Example – Step 1 and Step 2 – 13min30s. Link: `https://www.youtube.com/watch?v=hKcy_avxpJ0`
    - Video 3: RESafety Process and Example – Step 3 to Step 7 – 14min08s. Link: `https://www.youtube.com/watch?v=IB8LWmA73mg`
- TAM Questionnaire, implemented using the Google Forms platform. See Appendix B:
    - Page 1 – Introduction and email collection.
    - Page 2 – Consent terms regarding anonymity and data usage.
    - Page 3 – Personal profile questions.
    - Page 4 – RESafety evaluation questions (based on TAM).
- Supporting Materials:
    - Slide 1: Context and technologies involved. See Appendix C.
    - Slide 2: RESafety Process Illustrated. See Appendix D.
    - Safety Analysis of the Insulin Infusion Pump System. See Appendix E.

5.2.5 Validation of the Survey Instruments

To validate the evaluation instruments, a pilot study was conducted with postgraduate students from the Center for Informatics at the Federal University of Pernambuco. Four participants took part in this phase, which aimed to assess the clarity of the questionnaire items, the adequacy of the support materials, and the overall coherence of the evaluation procedure.

Based on the feedback received, minor adjustments were made — including the reduction of the duration of "Video 1: Introduction and Technology Overview" from 13 minutes and 13 seconds to 7 minutes and 20 seconds — to improve usability and engagement before the final evaluation.

5.2.6 Data Collection

The participants answered a questionnaire consisting of six demographic questions, twelve TAM-related questions using a 7-point Likert scale, and one open-ended question. Additionally, they were asked whether they would allow further contact if needed and, if so, to provide their contact information.

A total of 22 participants were invited via email, selected based on their expertise in requirements engineering and/or safety engineering. Of these, 14 confirmed their willingness to participate in the study. However, in the end 12 participants completed the survey. The study materials were then sent by email, and participants were instructed to access the official study website where the materials were made available. The expected completion time was set at 40 minutes, and the asynchronous nature of the study was emphasized. Participants had a total of 16 days to complete the survey.

5.2.6.1 Subjects Profile

Table 5.2 presents a summary of the demographic data of the 12 participants in the study. This is followed by an analysis of these data.

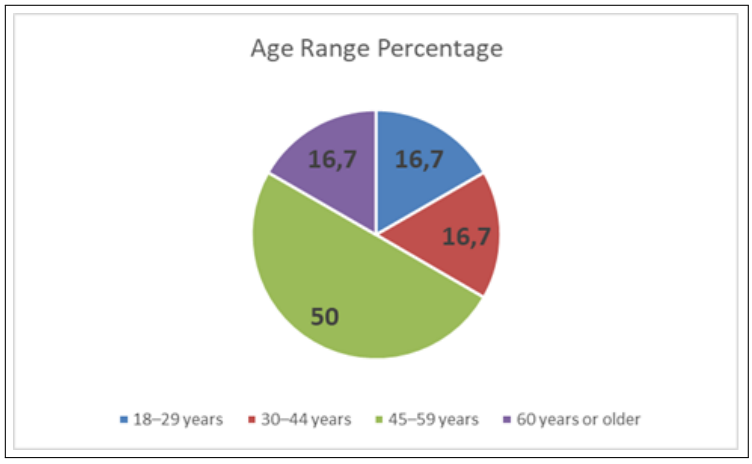
Table 5.2: Demographic profile of participants

ID	G	Age	Edu.	Area	Exp.
P1	F	30–44	Postdoc	RE	>10
P2	M	45–59	Postdoc	RE	>10
P3	M	45–59	Ph.D.	RE	>10
P4	M	60+	Postdoc	RE	>10
P5	F	18–29	Master	RE	3–5
P6	M	30–44	Master	RE	3–5
P7	M	45–59	Postdoc	RE	>10
P8	F	45–59	Postdoc	RE	>10
P9	M	60+	Postdoc	SE	>10
P10	M	18–29	Master	SE	6–10
P11	M	45–59	Ph.D.	RE	>10
P12	M	45–59	Ph.D.	RE	>10

Source: Author (2025).

As shown, the survey participants consisted of 25% Female (F) and 75% Male (M) respondents in terms of Gender (G). Regarding Age, the most prevalent group was between 45 and 59 years old, representing 50% of the participants. The remaining age groups—18–29, 30–44, and 60 or older—each accounted for 16.7% of the sample, as illustrated in Figure 5.1.

Figure 5.1: Age range of the survey participants



Source: Author (2025).

In terms of academic qualifications (Edu.), the majority (50%) held a *Postdoctoral Research/Fellowship* degree, while 25% held a *Doctorate/Ph.D. (Stricto Sensu)*, and 25% held a *Master's Degree (Stricto Sensu)*. All participants indicated that they reside in Brazil.

When asked about their primary professional or research area in the field of safety and/or requirements engineering (Area), 83.3% of participants (ten individuals) indicated *Requirements Engineering (RE)*, while only 16.7% (two individuals) selected *Safety Engineering (SE)*. It is important to note that, among the 10 invitees who did not participate in the survey, 7 were identified—based on an analysis of their publications—as having *safety engineering* as their primary area of expertise. Therefore, despite our efforts to achieve a balanced representation of both areas, this was not possible due to the lower participation of safety engineering specialists.

Regarding years of experience in their primary field, 75% of the respondents reported having more than 10 years of professional or research experience, and the remaining participants reported having a minimum of three years of experience.

Having thoroughly presented the methodology applied in our empirical study, we now turn to the analysis of the collected data. The aim of this section is to systematically address the three research questions defined earlier, namely: Survey – RQ1, Survey – RQ2, and Survey – RQ3.

5.3 Data Analysis

Table 5.3 summarizes the mean values and sample standard deviations for the responses to items Q01 to Q12 of the TAM questionnaire.

Table 5.3: Means and sample standard deviations of TAM items

Question	Mean	Sample Standard Deviation
Perceived Usefulness (PU)		
Q01	5.00	1.48
Q02	6.00	0.95
Q03	4.75	1.29
Q04	5.58	1.16
Q05	5.25	1.42
Q06	6.25	1.29
PU Mean (all items)	*5.47*	*1.26*
Perceived Ease of Use (PEOU)		
Q07	4.42	1.93
Q08	5.58	1.24
Q09	5.75	1.60
Q10	4.92	1.73
Q11	5.08	1.56
Q12	5.25	1.66
PEOU Mean (all items)	*5.17*	*1.62*

Source: Author (2025).

The analysis of the TAM questionnaire responses is presented in the following sections. The results are structured according to the two main constructs—Perceived Usefulness (PU) and Perceived Ease of Use (PEOU)—which correspond to questions Q01 through Q12, along with the insights obtained from the open-ended question. Since only 2 out of the 12 participants had their expertise defined as safety engineering, we will present the data considering this distinction.

5.3.1 Perceived Usefulness

Participants responded to six questions related to *Perceived Usefulness* (PU). As shown in Table 5.3, the overall mean across all participants was 5.47, with a sample standard deviation of 1.26. These results suggest that, in general, participants agreed that the RESafety process is useful, although there was some variability in the responses. The items related to this construct, listed in Table 5.1, correspond to questions Q01 through Q06.

The research question addressed in this section was:

Survey – RQ1 – How useful is the RESafety process for supporting the development of the safety analysis?

Table 5.4 provides a breakdown of the mean responses from participants who identified their primary field as either Requirements Engineering or Safety Engineering. The subsequent analysis discusses the main findings and implications derived from these results.

Table 5.4: Comparison of descriptive statistics (Mean and Sample Standard Deviation) for Q01–Q06 (Perceived Usefulness – PU) between Requirements and Safety Engineering participants

Code	Scale Item	Mean (Req.)	SD (Req.)	Mean (Saf.)	SD (Saf.)
Q01	Work More Quickly	5.20	1.48	4.00	1.41
Q02	Job Performance	5.60	1.08	4.50	0.71
Q03	Increase Productivity	5.60	0.70	5.00	1.41
Q04	Effectiveness	5.80	0.79	5.50	0.71
Q05	Makes Job Easier	5.70	0.67	5.00	0.00
Q06	Useful	6.00	1.00	5.50	0.71
–	**Overall (PU)**	**5.65**	**0.27**	**4.92**	**0.58**

Source: Author (2025).

As shown in Table 5.4, the average score for participants who reported **Requirements Engineering** as their primary area of expertise was **5.65**, with a **sample standard deviation of 0.27**. This low standard deviation among PU items suggests a high degree of agreement—participants consistently rated the usefulness of RESafety highly. The items with the highest variability in this group were **Q01 – Work More Quickly**, **Q02 – Job Performance**, and **Q06 – Useful**, each with a standard deviation equal to or greater than 1.

For the participants who identified **Safety Engineering** as their primary area—only two in this sample—the average score was **4.92**, with a sample standard deviation of **0.58**. While the overall perception of usefulness was lower in this group, the variability was slightly higher. However, the extremely small number of Safety Engineering respondents (two) makes this group's results statistically fragile, both when compared to the Requirements Engineering group and in general. Nevertheless, considering the participants' expertise, we chose to retain these responses as they provide meaningful insights.

It is worth noting that the item with the highest scores across both groups was **Q06 – Useful**, which asked: *"Overall, I find RESafety useful for safety analysis of Safety-Critical Systems."* This item received a mean score of **6.0** from the Requirements group and **5.5** from the Safety group. We interpret this as a strong indicator of the overall perceived usefulness of RESafety, supporting our hypothesis that the process is considered beneficial. Consequently, when aggregating all responses—that is, participants whose primary area of expertise was either requirements engineering or safety engineering—Q06 reached the highest overall mean of **6.25**.

In contrast, **Q01 – Work More Quickly** received the **lowest average scores**: **5.20** from Requirements Engineers and **4.00** from Safety Engineers, making it the lowest-rated item overall. It was also the item with the **highest variability**, with standard deviations of **1.48** and **1.41**, respectively. This result is not unexpected. Given that RESafety integrates two existing techniques, the process may initially demand more effort and time rather than accelerating tasks. For instance, users familiar with iStar4Safety must perform five additional steps involving STPA, while STPA users must learn two new steps and adapt their workflow to the iStar4Safety context.

Lastly, the **lowest variability** was observed in **Q05 – Makes Job Easier**, with standard deviations of **0.67** for Requirements Engineers and **0.00** for Safety Engineers. The corresponding mean scores were **5.7** and **5.0**, respectively. Considering the scale ranged from 1 (strongly disagree) to 7 (strongly agree), these results are generally positive. They suggest that both groups believe RESafety can help simplify the safety analysis process.

5.3.2 Perceived Ease of Use

Participants also answered six questions related to *Perceived Ease of Use* (PEOU), as presented in Table 5.4. The construct achieved, in an overall analysis of all responses, a mean score of 5.17 with a sample standard deviation of 1.62, slightly lower than the values obtained for the PU construct. These questions, previously presented in Table 5.1, correspond to items Q07 through Q12.

The main research question addressed in this section is.

Survey – RQ2 – How easy is it to use the RESafety process in the development of the safety analysis?

The table 5.5 below summarizes the data gathered for each question related to PEOU.

Table 5.5: PEOU – Mean and Sample Standard Deviation per Area (Q07–Q12)

Code	Scale Item	Mean (Req.)	SD (Req.)	Mean (Saf.)	SD (Saf.)
Q07	Easy to Learn	4.90	1.73	2.00	0.00
Q08	Controllable	5.70	0.95	5.00	2.83
Q09	Clear & Understandable	5.90	1.45	5.00	2.83
Q10	Flexible	5.00	1.49	4.50	3.54
Q11	Easy to Become Skillful	5.40	1.35	3.50	2.12
Q12	Easy to Use	5.40	1.35	3.50	2.12
–	**Overall (PEOU)**	**5.38**	**0.39**	**3.92**	**1.16**

Source: Author (2025).

As shown in Table 5.5, the average score for participants who reported **Requirements Engineering** as their primary area of expertise was **5.38**, with a **sample standard deviation of**

0.39. This low variability indicates a relatively consistent perception of ease of use among these participants, suggesting that they generally found the RESafety process to be accessible and straightforward. The item with the highest variability in this group was **Q07 – Easy to Learn**, with a standard deviation of **1.73**, possibly reflecting differing levels of prior familiarity with STPA or iStar4Safety.

In contrast, participants from the **Safety Engineering** group—again, only two in this sample—reported a lower average score of **3.92**, with a higher sample standard deviation of **1.16**. This suggests that these participants had a more critical or variable view regarding the ease of use of the RESafety process. Notably, the highest variability was observed in **Q10 – Flexible**, with a standard deviation of **3.54**, and in both **Q08 – Controllable** and **Q09 – Clear & Understandable**, each with a standard deviation of **2.83**. It is important to emphasize the small number of participants from the field of Safety Engineering (two), which makes these results statistically less reliable and should be interpreted with caution.

The most favorable item across both groups was **Q09 – Clear & Understandable**, which reached a mean of **5.90** among Requirements Engineers. Conversely, **Q07 – Easy to Learn** had the lowest score in the Safety group, at **2.00**, and was also the lowest across all PEOU items and also across the entire survey. This aligns with expectations, as the integration of iStar4Safety and STPA likely introduces a learning curve, especially for participants not previously familiar with one of the techniques.

The **lowest variability** in the Safety Engineering group was observed in **Q07 – Easy to Learn**, with a standard deviation of **0.00**, indicating a shared perception of difficulty in learning the process. Meanwhile, for the Requirements Engineering group, **Q08 – Controllable** exhibited the lowest variability, with a standard deviation of **0.95**, suggesting broad agreement among participants in responding positively to the statement: *"I find it easy to follow the steps of RESafety."*.

Overall, the results reveal that Requirements Engineers found the process relatively easy to use, while Safety Engineers exhibited more diverse and critical views. Despite the small number of participants in the Safety Engineering group, their insights remain valuable due to their domain expertise and help us identify areas for potential improvement in training.

5.3.3 Improvements Suggested for the Process

The participants were invited to provide additional insights, encouraging them to suggest improvements and/or adjustments they deemed necessary to the RESafety process. In this section, we discuss the most relevant insights obtained from responses to the question: *"Do you have any additional remarks? Please write them below. Your opinion is very important to our research."*

It is important to note that no formal coding analysis was performed on the responses, as this question was optional and intended solely to gather general opinions from participants who wished to contribute them. Therefore, we report and analyze here the responses that were

provided, classifying them according to their intent.

In Appendix G, which contains the open-ended responses from the survey conducted with experts in requirements engineering and safety engineering, we include all the responses translated into English, organized by participant. Participant **P12** did not respond to this question.

Below, we present a dedicated analysis for each identified category:

- Strengths of RESafety
- Limitations / Suggestions
- Methodological Considerations

We would like to emphasize that participants **P9** and **P11** were the ones who identified their primary area of expertise as **Safety Engineering**.

5.3.3.1 Strengths of RESafety

Table 5.6 presents a summary of responses classified under the category "Strengths of RESafety."

Table 5.6: Open-ended Responses – Category: Strengths of RESafety

ID	Statement	Participant	Excerpt (in English)
S1	Integration of modeling and safety concepts	P1	"One of the strengths of the process is the way it integrates well-established modeling techniques (such as iStar) with safety analysis concepts, such as control structures. The iterative nature is also an important aspect, as it allows for progressive refinement, which aligns with real-world software development cycles.[. . .]"
S2	Valuable contribution to RE and safety	P1	“Overall, I consider this methodology to be a valuable contribution to the field of requirements engineering focused on the safety of critical systems.”
S3	Well documented and easy to follow	P3	“The process is well documented and easy to follow.”
S4	Use of BPMN flow and artifacts improves understanding	P6	"The presentation of the RESafety process as a BPM flow, detailing each subprocess and highlighting the input and output artifacts, are well-founded decisions, as they greatly support the understanding of each step and of the overall process."
S5	No difficulties in understanding and integration with STPA	P6	“[. . .]as a Requirements Engineer with experience in STPA, I encountered no major difficulties in understanding the process and the integration between the approaches.”
S6	Increased expressiveness through technique combination	P6	“[. . .]I see potential in the combination of these techniques to enhance the expressiveness of safety analysis by adding important and complementary information.”

Source: Author (2025).

As shown, this category includes responses from three experts in requirements engineering and one from the two experts in safety engineering. The most frequently cited advantageous feature was the process documentation (S3, S4), particularly highlighted through BPMN modeling. Given the number of steps and the variety of tasks involved, we prioritized demonstrating our process using BPMN process modeling to clearly illustrate the artifacts, tasks, and their

sequencing in a temporal manner. Another aspect that received praise was the combination of the two techniques (S1, S6), which represents the primary goal of this work. The remaining statements (S2, S5) referred to RESafety as a valuable contribution and mentioned that there were no difficulties in understanding the process and the integration between the approaches, respectively.

5.3.3.2 Limitations / Suggestions

We now present in Table 5.7 the limitations and/or suggestions provided by the experts regarding the RESafety process.

Table 5.7: Open-ended Responses – Category: Limitations / Suggestions

ID	Statement	Participant	Excerpt (in English)
S1	Need for case study	P2	"Therefore, I believe it is essential that, following this static evaluation based on a questionnaire, a real case study be conducted, and that the results of the case study be subsequently compared with the findings of this questionnaire-based study"
S2	Familiarity with iStar is a barrier	P3	“[. . .]This might represent a challenge for adopting the approach, given that many practitioners are not familiar with or do not use iStar.”
S3	Missing dependency analysis	P4	"From what I observed, I believe a part is missing where obstacles originate from dependencies. Based on the video, it seems that the cases are analyzed within each actor, but what about those derived from dependencies?"
S4	Need for better guidance materials	P5	“Inexperienced users may abandon the tool out of frustration if there is no well-structured and detailed guidance material to support them”
S5	Control structure requires high expertise	P5	“Correctly modeling the control structure demands knowledge in systems engineering, automation, or software architecture, which can be a significant barrier for requirements engineers.”
S6	Unclear practical value of iStar4Safety in a step	P5	“It is not clear how following the istar4safety model could support this step in practice.”

Continued on next page

Table 5.7 – *continued from previous page*

ID	Statement	Participant	Excerpt (in English)
S7	Improve artifact integration visualization	P6	"Still regarding the integration between iStar4Safety and STPA [...] it would be helpful to further highlight these connections and the relationships between the related artifacts in the process, perhaps by using some diagram or schematic, in addition to the BPM process itself, to illustrate the integration."
S8	Evaluate adoption by different backgrounds	P6	"[. . .]It is also important to evaluate its potential for use by requirements engineering specialists (with or without experience in safety) who are not familiar with iStar and/or STPA, in order to analyze how professionals with different backgrounds perceive the proposed process."
S9	Need for tool support	P6	"[. . .] I believe that the development of a tool to support the execution of the process could greatly contribute to its adoption."
S10	Missing SPEM 2.0 modeling	P7	"I missed a modeling of the process using the SPEM 2.0 notation[. . .]"
S11	Domain expert support not included	P7	"Shouldn't there at least be the support of a domain expert specified in the process?"
S12	Unclear modeling construct extension	P7	"It was not clear whether there was an extension of the iStar modeling constructs to represent safety elements from the STPA perspective."
S13	Include common mistakes and mitigation	P7	"A list of likely mistakes that safety analysts might make when using your approach, along with guidance on how they could prevent them."
S14	Uncertainty about optional steps	P7	"Are steps 3.6 and 3.7 optional? There may be no sensors or actuators (video 2)."
S15	No explanation of model links in video	P8	"In the video, there is no explanation of the semantics of the link between «SafetyGoal» SC and «SafetyGoal» R (Step 2), nor of the relationships among UA, LS, and SR (Step 7)."
S16	Lack of background in i* hinders understanding	P8	"People who are not familiar with i* may lack the appropriate background to fully understand the explanations in the video."

Continued on next page

Table 5.7 – *continued from previous page*

ID	Statement	Participant	Excerpt (in English)
S17	Need to highlight benefits of i* extension	P8	"It is necessary to highlight the benefits of using the extended i* model."
S18	Risk of increasing complexity and STPA issues	P9	"The proposed process runs the risk of requiring additional analysis, thereby increasing its complexity. Furthermore, there are some misunderstandings in the application of the STPA steps."
S19	Consider maturity levels for application	P11	"[. . .]What could be considered by the research is the issue of maturity levels required to use the proposed process."

Source: Author (2025).

As can be seen, participant P2 (S1) highlighted the need for modeling a real use case study. Although we believe this would have greatly enriched our thesis, there was not enough time to carry out such a study. To address this threat with the available resources, we modeled a real system — a robotic arm-based medication delivery system deployed in a public hospital (see Chapter 4). Additionally, P3 (S2), P5 (S4), and P8 (S16) raised concerns about the potential limitation imposed by unfamiliarity with the underlying approaches. We acknowledge this issue; however, we sought to mitigate it through a combination of step-by-step explanations, BPMN modeling, and the applied example in the IIPS context.

Participant P4 (S3) pointed out the absence of safety elements such as dependencies in iStar. This limitation stems from the fact that, during the creation of iStar4Safety , we established that dependency elements could not represent safety elements (Ribeiro *et al.*, 2019a), as it is not feasible to model the entire safety logic tree for such elements. As highlighted by Leveson (1995), it is essential to model the complete safety logic up to its mitigation.

Participant P5 pointed out that the use of the control structure concept may require high expertise (S5). They also mentioned not seeing the use of iStar4Safety for this step (S6). Indeed, we acknowledge that the concept of control structure, although presented in this thesis and in the RESafety process, should be further assimilated through classical materials such as the STPA Handbook (Leveson & Thomas, 2018), as it may not be familiar to or commonly used by requirements engineers. However, when integrating approaches from two distinct areas—requirements engineering and safety engineering—it is expected that certain elements unfamiliar to one of the domains will need additional clarification. Regarding the use of the iStar4Safety model, we argue that by adopting a high-level requirements modeling approach, we enable a system-wide representation in Step 3 and throughout the entire analysis, without losing critical details typically found only in lower-level specifications of actors and control

actions Moreover, as can be seen in the response of participant P10 in Appendix G, they indicated that the iStar4Safety models may offer advantages by providing information that supports the accurate modeling of systems and may be particularly useful for systems with well-defined actors from the outset.

Participants P6, P7, and P8 highlighted the need to establish clearer connections between the artifacts of the two approaches (S7, S12, S15). The detailed explanations were not included in the video due to time constraints, as the presentation needed to remain feasible for evaluation purposes. However, we believe this concern was mitigated in Chapter 3 of this thesis.

Participant P6 pointed out the need for an evaluation involving experts with or without prior knowledge in safety (S8). We acknowledge that this is an essential step; however, it is planned as future work. The development of a tool to support the use of our approach is also considered future work. We emphasize that such development was initiated (S9), but due to time constraints, it was not completed.

Participant P7 pointed out the need for modeling using SPEM 2.0 (S10). However, modeling our process using BPMN was a design decision. Regarding the inclusion of a domain expert during modeling (S11), we emphasize that this is indeed recommended by the authors of STPA as a step that facilitates the analysis process, and this should not be different for our approach. Nevertheless, we believe that the definition of the roles played by analysts in the process is up to the development team. The participant also mentioned the importance of including possible errors and mitigation strategies when using our approach (S13), which we consider as future work. As for the uncertainty about the optional steps (S14), elements from Steps 3.3 to 3.8 were modeled as optional in the BPMN, as shown in Figure 3.10.

Participant P8 pointed out the need to highlight the benefits of using iStar4Safety (S18). As discussed in previous chapters, iStar4Safety brings a GORE-based perspective to the modeling process, allowing users to represent high-level initial requirements through a strategic social relationship language among actors. This high-level view supports knowledge management and helps ensure that essential elements are not lost in more specialized views during the early stages of the analysis.

Finally, participant P11 pointed out the need to consider maturity levels (S19). We believe that this perspective was not within the scope of our current work. However, we will consider this need for future projects.

5.3.3.3 Methodological Considerations

Finally, we present the last category, *Methodological Considerations*, in Table 5.8.

Table 5.8: Open-ended Responses – Category: Methodological Considerations

ID	Statement	Participant	Excerpt (in English)
S1	Limitation of evaluation based only on videos	P2	"Answering the questionnaire based solely on the videos, without having used the process in a real case, may limit the evaluation."
S2	Difficult to evaluate performance without real use	P3	“[. . .]It is difficult to assess its performance without using it in practice.”
S3	Trade-offs depend on application context	P10	“In summary, when choosing between applying ’pure’ STPA or RESafety, it seems that the advantages and disadvantages of each will depend on the specific application cases.”

Source: Author (2025).

We classified three statements in this category. Participants P2 and P3 pointed out the difficulty of evaluating the approach without actually using it (S1, S2). We acknowledge that this represents a gap in our evaluation. However, conducting an evaluation that required actual use of the approach would demand significantly more training time and participant availability. Our decision to carry out the evaluation solely through tutorials and videos, despite the associated threats, was based on the cost of involving participants who could provide valuable insights into our process. As future work, we intend to conduct another evaluation that enables the assessment of the process through its actual application.

Regarding S3 sentence, provided by participant P10, an expert in safety engineering, we would like to add the remainder of the response:

> "[...]For example, one of the biggest difficulties I encounter when performing STPA is obtaining information for accurately modeling systems, especially those already built. In a system with well-defined actors from the beginning, the SD and SR models can be quite helpful in this stage. But for systems with "more abstract" actors, it seems to me that the difficulty would remain (I’m thinking about my reality of applications in oil extraction, where a large part of the information required for a detailed model ends up being confidential)."

We therefore conclude that the inclusion of iStar4Safety in our proposed process contributes meaningfully to the modeling and execution of safety analyses. Its goal-oriented, high-level abstraction supports the identification and clarification of system elements early in the process. This broader perspective enables analysts to capture the structure and interactions of

the system comprehensively, reducing the risk of overlooking relevant aspects as the analysis evolves to more detailed levels.

In the following section, we present the improvements and adjustments made to the RESafety process based on the feedback obtained during the evaluation.

5.3.4 Response actions and Process Improvements

Below, we present a series of actions and improvements made to the RESafety process, inspired by our evaluation and the insights provided by the participating experts. It is important to note that some items will be addressed as future work due to time constraints, and these will be explicitly indicated.

- Modeling of our approach using a real system (presented in Chapter 4).
- Explanation of the process through the combination of step-by-step descriptions, BPMN modeling, and the applied example in the IIPS context.
- Clarification of the connections between the elements of the approach (detailed in Chapter 3).
- Additional evaluation with requirements engineering professionals both with and without safety knowledge, as well as with safety professionals both with and without requirements engineering knowledge (Future work).
- Development of a tool to support the proposed process (Future work).
- Creation of a guideline demonstrating possible errors and corresponding mitigation strategies (Future work).
- Investigation of the need to address different safety maturity levels (Future work).
- Evaluation of RESafety through empirical studies that involve practical use of the technique by participants (Future work).
- Creation of a guideline identifying favorable contexts for applying RESafety instead of using the techniques separately (Future work).

Next, we present the threats to the validity of this study.

5.3.5 Validity threats

In order to address the trustworthiness of this study and demonstrate the extent to which its results are valid and not biased by our own perspective, we present the threats to the validity of our evaluative study, based on the approach proposed by Wohlin *et al.* (2012).

5.3.5.1 Conclusion Validity

Conclusion validity concerns the ability to draw accurate conclusions about the relationship between the treatment and the outcomes of the study.

- *Low statistical power:* This item aims to assess the statistical power to reveal true patterns in the data. Considering that the sample size in this study included only twelve participants, with an uneven distribution (ten requirements engineers and only two safety engineers), the statistical power is below the ideal threshold. To mitigate this threat, we presented comparative analyses between the two groups and consistently highlighted the number of participants from each area.

- Random irrelevancies in experimental setting: This threat refers to elements outside the experimental context that may nonetheless influence the outcomes. Since participants were allowed to complete the evaluation in any location of their choice, this threat is present in our study, as we had no control over the environment or the amount of time each participant dedicated to the evaluation.

- *Random heterogeneity of subjects:* The heterogeneity of participants can pose a risk to conclusion validity. In our study, we initially aimed for a more homogeneous group in terms of primary area of expertise. However, as shown, there was limited participation from safety engineers. Another source of heterogeneity was gender, with a predominance of male participants. On the other hand, the highest level of education and years of experience in the primary area were concentrated within only three ranges—3 to 5 years, 6 to 10 years, and more than 10 years—suggesting a more homogeneous profile in these aspects.

5.3.5.2 Internal Validity

Threats to internal validity can compromise the integrity of the independent variable by interfering with the ability to establish a causal relationship, thus weakening the conclusion about the connection between the treatment and its outcomes. Since this survey did not include a control group, certain threats may have influenced the results as a consequence. Therefore, this study considered the following single-group internal validity threats:

- *History:* This threat refers to the possibility of different treatments being applied at different points in time during the study. The treatment received by participants may be influenced if it is administered at varying moments. This threat was present in our study due to its asynchronous nature. However, making the evaluation both individual and asynchronous also helped mitigate this issue by allowing participants to choose the most appropriate moment to complete it.

- *Maturation:* This threat concerns how participants may change or react over time, potentially affecting the study outcomes. To address this threat, we designed concise videos and a focused questionnaire to minimize the time required for completion. Additionally, we provided the instructional materials in both video and slide formats, allowing individuals to choose the format that best suited their preferences.
- *Instrumentation:* This threat refers to effects caused by the instruments used in the study. To address it, we conducted a pilot study with four postgraduate-level participants who provided feedback on the instruments and offered valuable insights for improvement.
- *Selection:* This threat relates to how participants are chosen for the study. In our case, we sought to select individuals with knowledge in the related fields of safety and requirements engineering to form a more heterogeneous group. However, this remains a bias in our study, as the number of safety engineers was small due to low participation from this group.
- *Mortality:* This threat concerns the dropout rate of participants during the experiment. As previously mentioned, we lost potential participants from the safety engineering domain, which introduced a significant bias to our results.

5.3.5.3 Construct Validity

Construct validity concerns the extent to which the results of a study can be generalized to the underlying theoretical constructs. Some threats are related to the study design, while others stem from social factors.

- *Design threats:* These threats are related to how well the study design reflects the intended constructs.
    - *Mono-operation bias:* This threat refers to the use of a single independent variable, subject, or treatment, which may limit the completeness of the results. In our evaluation, although only one treatment was applied, we emphasize its complexity, as it involves the integration of two different technologies. Implementing multiple treatments would have required significantly more time to train participants, including instruction in related topics. Therefore, we believe the chosen approach was the most feasible. Additionally, to mitigate this threat, we used the TAM model in the questionnaire, which is designed to assess the perceived usefulness and ease of use of the evaluated technology.
    - *Confounding constructs and levels of constructs:* This threat refers to differences in understanding or expertise across the constructs involved. To

address it, we provided a tutorial video explaining the applied approaches, as well as a detailed BPMN model illustrating the step-by-step process for artifact creation—encompassing both iStar4Safety and STPA. Moreover, only one participant reported having in-depth knowledge of both approaches. Unfortunately, we were unable to find more participants with balanced expertise in both domains.

- *Social threats:* These threats are related to participants' behavior during the study.
  - *Hypothesis guessing:* Participants may try to infer the purpose of the study and adjust their responses or behavior to intentionally support or undermine its expected outcomes. To mitigate this effect, participants were encouraged to respond impartially and were informed that their participation would be treated anonymously.
  - *Evaluation apprehension:* Participants may feel uneasy or self-conscious about being evaluated. To address this threat, we emphasized the anonymous and voluntary nature of their participation throughout the study.

5.3.5.4 External Validity

Finally, we address the threats to external validity. These threats concern the limitations in generalizing the results of the experiment to industrial practice. The following threats are highlighted:

- *Interaction of selection and treatment:* This threat concerns the risk of having non-representative subjects participating in the experiment. In our case, we sought to include both types of participants—those with knowledge in requirements engineering, in safety engineering, or in at least one of the two areas. Additionally, we collected data on their professional experience, and all participants had at least three years of experience in their primary area of expertise.
- *Interaction of setting and treatment:* This threat refers to the risk that the materials and setup used in the study do not reflect real-world industrial practice. In our case, we used a commonly referenced system in the literature—the Insulin Infusion Pump. We aimed to describe it using reliable and realistic data. However, it was necessary to adopt it as a toy example due to the limited time available for the evaluation. Moreover, given the number of steps in our approach, using a smaller example allowed participants to better understand the technique and follow its execution. Nonetheless, we believe that the material used in the experiment

adequately represents a simplified version of what would be found in real-world practice.

5.4 Chapter summary

This chapter presented the evaluation of the RESafety process using a questionnaire based on the Technology Acceptance Model (TAM), with the participation of experts in Requirements Engineering and Safety Engineering. The objective of the evaluation was to understand participants' perceptions regarding the *Perceived Usefulness (PU)*, *Perceived Ease of Use (PEOU)*, and the overall applicability of the RESafety process.

The results revealed a generally positive perception. The PU scores indicated that participants recognized the potential benefits of the process for safety-critical systems. Although the PEOU scores were slightly lower, they reflected that certain steps of the process may be complex or demand a higher learning curve—especially due to the integration of two distinct techniques: iStar4Safety and STPA. Nonetheless, several participants emphasized the quality and clarity of the documentation and process modeling used to explain the methodology, including BPMN diagrams and applied examples.

The open-ended responses were analyzed through three categories: **Strengths of RESafety**, **Limitations / Suggestions**, and **Methodological Considerations**. Positive feedback included recognition of the comprehensive documentation, iterative nature of the process, integration between the techniques, and the clarity of the step-by-step explanation. Constructive suggestions included the need for: (1) deeper explanation of artifact integration; (2) development of supportive tools; (3) consideration of safety maturity levels; and (4) evaluations based on real-case applications. Some experts also noted the difficulty of assessing the process without hands-on experience, emphasizing that future work should include empirical studies involving the actual use of the process.

Based on this evaluation, several enhancements and future directions were identified and documented, many of which are already being considered for the continuation of this research.

The next chapter presents the final conclusions of this work, addressing the research questions initially defined, highlighting the main contributions, discussing the study's limitations, and proposing directions for future research.

6

Conclusions

Safety-critical systems are generally complex and non-trivial, composed of multiple interacting components with intricate dependencies. Analyzing such systems—especially in terms of safety—is itself a non-trivial task that requires particular care from the earliest development stages. This is crucial to avoid accidents that could result in significant financial losses, environmental damage, or even loss of human life.

In response to these challenges, this thesis proposed RESafety, a process designed to support the early modeling of safety requirements by integrating iStar4Safety goal models with the STPA safety analysis technique, which is grounded in systems theory.

At the core of this research lies the following problem: *How can Requirements Engineering and Safety Engineering be systematically integrated—through a goal-oriented approach and a system-theoretic safety analysis technique—to support the early modeling and traceability of safety requirements in safety-critical systems?* This question guided the development of all stages of this work, from the selection of techniques to the definition, application, and evaluation of the proposed process.

To address this problem, an literature review was conducted to identify GORE languages and safety analysis techniques suitable for integration. Previous studies had indicated a lack of GORE languages capable of fully modeling the constructs necessary for early safety analysis (Vilela *et al.*, 2017b). Consequently, we concluded that iStar4Safety, previously proposed in Ribeiro *et al.* (2019b), was an appropriate GORE language for modeling safety requirements. In parallel, studies have demonstrated that STPA can identify all hazard scenarios typically found by techniques such as HAZOP (McDermid *et al.*, 1995) and FTA (Ruijters & Stoelinga, 2015), and even additional ones. Moreover, STPA is applicable across various development phases—from system architecture to detailed design and implementation—making it the most suitable safety analysis technique for our purposes.

Initial attempts to integrate iStar4Safety and STPA involved using intermediate modeling frameworks such as UCM Ribeiro *et al.* (2023a) and BPMN Ribeiro *et al.* (2023b) to facilitate the connection between the two approaches. However, in a subsequent study Ribeiro *et al.* (2024), we succeeded in defining a process that relies solely on iStar4Safety and STPA. We further

refined this process by adjusting specific steps—particularly those related to loss scenarios and safety requirements—to complete the logical flow of the safety analysis with elements derived from STPA.

The final version of the RESafety process was defined based on insights from both techniques, requiring only minor adaptations to their original structures. The resulting process comprises six iterative steps, each producing specific artifacts that collectively support the systematic modeling and traceability of safety requirements. To enhance clarity and comprehensibility, the process was modeled using BPMN. We first illustrated its application using the well-known Insulin Infusion Pump (IIP) example from the literature and then applied it to a real-world case involving a Robotic Arm-Based Medicine Delivery System operating in a hospital pharmacy setting.

Finally, to assess the practical relevance and acceptance of the proposed process, an empirical evaluation was conducted using the Technology Acceptance Model (TAM). The evaluation focused on two key constructs—Perceived Usefulness (PU) and Perceived Ease of Use (PEOU)—and included an open-ended question to gather qualitative feedback from experts in Requirements Engineering, Safety Engineering, or both. Although most participants identified Requirements Engineering as their primary area of expertise, the collected feedback was highly valuable. The evaluation was conducted through a dedicated website that provided tutorials, supporting materials, and the evaluation questionnaire. Overall, the results revealed a positive perception of RESafety, with perceived usefulness being rated slightly higher than ease of use. The feedback was categorized and analyzed, leading to concrete improvement actions for the process.

In the following section, the research questions that guided this work are revisited, and their corresponding answers are presented.

6.1 Answering Research Questions

We now present the answers to the research questions defined in Section 1.3 of this thesis.

6.1.1 Research Question 01

Which Goal-Oriented Requirements Engineering (GORE) language and safety analysis technique are suitable for supporting the modeling of safety requirements in safety-critical systems?

To address RQ01, an extensive literature review was conducted to identify Goal-Oriented Requirements Engineering (GORE) languages and safety analysis techniques suitable for modeling safety requirements in safety-critical systems. This investigation aimed to understand not only which approaches exist but also how effectively they support the systematic derivation,

representation, and traceability of safety-related artifacts.

Previous studies Vilela *et al.* (2017b) revealed that traditional GORE languages, such as *iStar* and *KAOS*, provide strong support for early requirements elicitation and rationale representation. However, they exhibit limitations in expressing safety-specific constructs, such as hazards, unsafe control actions, and safety constraints. This lack of expressiveness compromises the alignment between requirements and safety perspectives—an essential aspect in safety-critical contexts where the omission or misrepresentation of safety elements may lead to incomplete or inconsistent specifications.

Conversely, extensions of goal-oriented languages, such as **iStar4Safety** -proposed by Ribeiro *et al.* (2019b) - have demonstrated potential to bridge this gap by incorporating constructs specifically designed to represent safety concerns. iStar4Safety introduces elements such as *Safety Goals*, *Hazards*, *Safety Tasks*, and *Safety Resources*, which can be associated with traditional intentional elements (goals, tasks, and dependencies). This integration allows safety-related aspects to be represented within the same modeling framework used for system goals, enhancing both expressiveness and traceability. The empirical studies conducted in earlier works (e.g., Ribeiro (2019)) confirmed that iStar4Safety improves analysts' ability to reason about safety during early system modeling, reinforcing its adequacy for integration into a structured process such as RESafety.

Regarding safety analysis techniques, the literature indicates that the **System-Theoretic Process Analysis (STPA)** stands out for its ability to conceptualize safety as a control problem rather than a mere reliability issue. Traditional methods such as FTA, FMEA, and HAZOP tend to focus on component-level failures and assume linear causality, which are insufficient to capture the complexity of modern socio-technical systems. In contrast, STPA—grounded in Leveson's **STAMP** framework—supports the identification of *Unsafe Control Actions (UCAs)*, *Loss Scenarios (LSs)*, and *Safety Constraints (SCs)* based on system interactions, control loops, and feedback mechanisms. This systemic perspective aligns naturally with the relational and dependency-oriented nature of GORE models, making STPA a strong candidate for integration with goal-oriented approaches.

The integration of iStar4Safety and STPA thus emerged as a theoretically consistent and empirically justified choice. While iStar4Safety provides a means to represent stakeholder intentions, dependencies, and responsibilities in a structured and traceable manner, STPA offers a rigorous analytical framework to identify and reason about hazards, unsafe interactions, and mitigation strategies. Their combination enables the systematic derivation of safety requirements from safety analysis results, ensuring bidirectional traceability between safety artifacts and stakeholder goals.

This finding directly informed the design of the **RESafety process**, whose iterative structure (presented in Chapter 3) operationalizes this integration. iStar4Safety is employed to model the intentional and organizational dimensions of the system at the goal-oriented level, whereas STPA supports the analytical identification of unsafe control actions and their corresponding

safety constraints. The synergy between these techniques was demonstrated in the analysis of the *Insulin Infusion Pump System (IIPS)* and the *Robotic Arm-Based Medication Delivery System*, where their combined use enabled the derivation of numerous safety requirements that would otherwise remain implicit or fragmented.

In summary, RQ01 is answered by establishing that **iStar4Safety** is the most suitable GORE language and **STPA** the most appropriate safety analysis technique for supporting the modeling of safety requirements in safety-critical systems. Their complementary strengths—expressiveness for early requirements modeling and analytical depth for hazard identification—provide a robust and theoretically grounded foundation for integrating Requirements Engineering and Safety Engineering, as embodied in the RESafety process.

6.1.2 Research Question 02

How can elements of Goal-Oriented Requirements Engineering models be systematically interrelated with those from safety analysis techniques to support integrated safety modeling?

To address RQ02, an in-depth analysis was conducted to determine how elements from Goal-Oriented Requirements Engineering (GORE) models—particularly those of **iStar4Safety**—could be systematically interrelated with the artifacts generated by the safety analysis technique **STPA**. The objective was to establish a coherent and traceable integration that enables the combined modeling of functional and safety aspects within a unified process.

Initially, the conceptual alignment between the two techniques was examined. iStar4Safety, as an intentional modeling language, represents *goals*, *tasks*, *resources*, and *qualities*, complemented by safety-oriented constructs such as *Hazards*, *Safety Goals*, *Safety Tasks*, and *Safety Resources*. STPA, in turn, is structured around the identification of *losses*, *system-level hazards*, *unsafe control actions (UCAs)*, *loss scenarios*, and *safety constraints* derived from the control-theoretic model of the system. Despite their different natures—iStar4Safety being declarative and social, and STPA being analytical and process-oriented—both share the common objective of supporting reasoning about system safety from the early stages of development.

The literature review and subsequent analytical comparison indicated that this interrelation could be established directly, without the need for intermediary modeling layers or transformations. To operationalize this integration, correspondences were defined between iStar4Safety and STPA elements. Table 6.1 presents the association between these elements. This mapping provided the foundation for a traceable and bidirectional relationship between stakeholder intentions and safety artifacts, enabling safety concerns identified in STPA to be reflected in the intentional models of iStar4Safety and vice versa.

Table 6.1: Mapping of RESafety artifacts and their representation in iStar4Safety and STPA.

RESafety (Documents)	**RESafety (Models)**	**iStar4Safety**	**STPA**
Components	Actors, agents, roles	Actors, agents, roles	Controller / Controlled processes (Hierarchical control structure)
Losses (Table in Step 1)	Not applicable	(indirectly) negation of a Safety Goal	Losses
Hazards (Table in Step 1)	Not applicable. (ID associated to Safety Constraint — SG — in SR model)	Hazards	System-Level Hazards
System-Level Constraints (Table in Step 1)	Safety Goal (ID: SC–X); refines actor main goal	Safety Tasks	System-Level Constraints
Responsibilities	Safety Goal (ID: R–X); refines the constraint it fulfills	Safety Goal	Responsibilities
Other non-safety elements	Goals, tasks, qualities, dependencies, links	Goals, tasks, qualities, dependencies, links	Not applicable
Control actions (Hierarchical control structure, Step 3)	Actor dependencies in SD model (may be adapted)	Not applicable	Control Actions
UCA table (Step 4)	Hazards (ID: UCA–X); linked via OBSTRUCT to the endangered Responsibility	Hazards	Unsafe Control Actions
Hazardous Conditions table (Step 4)	Hazards (ID: HC–X); linked via OBSTRUCT to the endangered Responsibility	Hazards	Not applicable
Controller Constraints table (Step 4)	Not applicable	Not applicable	Controller Constraints
Loss Scenarios table (Step 5)	Hazards (ID: LS–X); linked via REFINEMENT to the causal UCA/HC	Hazards	Loss Scenarios
Safety Requirements (Step 5)	Safety Tasks (ID: SR–X); linked to mitigated Loss Scenarios	Safety tasks	Not applicable

Source: Author (2025).

Building upon this conceptual foundation, the **RESafety process** was defined as a structured framework to guide this integration. The process—comprising six iterative steps—supports the identification of general and safety concerns, the modeling of iStar4Safety diagrams, the construction of the control structure, the identification of unsafe control actions, safety constraints, and loss scenarios, the derivation of safety requirements, and the update of the intentional models. Each step was formally represented using **BPMN** to enhance transparency, repeatability, and understanding of the workflow, especially concerning the interactions between the safety and requirements perspectives.

The integration was validated through two applications. The first, involving an *Insulin Infusion Pump (IIP) System*, demonstrated the feasibility of applying RESafety to a well-documented safety-critical domain. The second, a real-world *Robotic Arm-Based Medication Delivery System*, allowed the process to be tested in a socio-technical environment that combined human and robotic interactions. In both cases, the systematic interrelation of iStar4Safety and STPA enabled the derivation of comprehensive sets of safety requirements, the preservation of

traceability between hazards and goals, and a deeper understanding of the rationale underlying safety decisions.

From this analysis, it is concluded that a systematic integration between GORE models and safety analysis techniques is feasible when guided by a process such as RESafety, which explicitly defines how artifacts from each domain interact and evolve iteratively. The integration demonstrates the complementary strengths of iStar4Safety—its expressiveness for representing stakeholder intentions and safety goals—and of STPA—its rigor in identifying hazards and deriving safety constraints. Together, these techniques enable an integrated safety modeling approach that supports consistency, completeness, and traceability across requirements and safety artifacts throughout the early development of safety-critical systems. Moreover, the process promotes the creation of artifacts at each of its steps, ensuring that all elements relevant to the analysis are properly documented and systematically maintained.

6.1.3 Research Question 03

What are the perceived usefulness, perceived ease of use, and expert-suggested improvements regarding the RESafety process?

The results indicated a positive overall perception of the RESafety process. Quantitatively, Perceived Usefulness received slightly higher ratings than Perceived Ease of Use, suggesting that experts recognized the method's capacity to add value to the use of RESafety and its resulting artifacts, even though some steps demanded a steeper learning curve. The process was perceived as capable of supporting systematic safety reasoning and maintaining traceability between hazards, goals, and safety constraints—key advantages particularly noted by participants experienced with safety analysis techniques such as STPA.

Qualitative feedback reinforced these findings. Experts highlighted the methodological coherence and structured nature of the process but pointed out challenges related to tool support, modeling complexity, and the need for further improvements. Suggestions included automating traceability among artifacts, simplifying the representation of STPA results, evaluating the process through an additional case study, and developing an integrated environment to visualize dependencies between iStar4Safety and STPA elements. These recommendations were categorized and analyzed, leading to concrete improvement actions such as refining the description of the process steps, merging redundant activities, and enhancing clarity in the BPMN representations. Other aspects mentioned by the experts were identified as directions for future work, including extending the evaluation to different domains, improving tool integration, and exploring automation strategies to support process execution.

In summary, the TAM-based evaluation confirmed that the RESafety process is perceived as both useful and reasonably easy to use, with its usefulness being particularly recognized in promoting alignment between Requirements Engineering and Safety Engineering perspectives.

The feedback provided by experts was instrumental in guiding the refinement of the process, ensuring that subsequent iterations of RESafety offer greater usability, reduced modeling effort, and improved analytical traceability across safety artifacts.

6.2 Contributions

RESafety is a novel approach that supports the early modeling of safety requirements by integrating iStar4Safety goal models with STPA safety analysis technique, which is grounded in systems theory. The main contributions of this thesis are presented below:

- **Identification of a suitable GORE language and safety analysis technique for integration.** The identification of iStar4Safety goal modeling language and STPA safety analysis technique as appropriate approaches for integration.
- **The definition of the RESafety Process.** It enables the modeling of requirements from a high-level, goal-oriented perspective using iStar4Safety, while also supporting the identification of safety-related elements through steps based on the STPA technique.
- **The illustration of the use of the RESafety process in a real world context.** We developed the first iteration of the integration of requirements modeling and safety analysis for a robotic arm responsible for delivering medications within a hospital pharmacy. It can be reused and refined for real-world scenarios where similar systems are deployed.

6.3 Limitations

Despite all our efforts, this work presents some identified limitations:

- **Lack of a dedicated tool to support semi-automated adoption.** We did not propose a tool to support the semi-automated application of our approach. We acknowledge that the absence of such a tool is a significant barrier to broader adoption. However, we attempted to mitigate this limitation by indicating existing tools that can be used in each step of the process and by providing the modeled examples as references.
- **Absence of a formal case study application.** A formal case study was not conducted due to time constraints associated with completing the final version of this work. However, this limitation was mitigated by modeling a real safety-critical system in which the authors were directly involved during its development, ensuring the applicability and realism of the proposed approach.

- **Challenges in adopting RESafety in real development environments for Safety-Critical Systems (SCS):** Applying the RESafety process in real-world development environments for Safety-Critical Systems presents several challenges. First, understanding the approach requires foundational knowledge of both iStar4Safety (or at least iStar) and STPA, as noted by experts during the evaluation.

  Although tutorial videos and explanatory materials in multiple formats—natural language, BPMN notation, and illustrative examples of two systems—were provided, beginner users may still need to consult additional references or introductory courses to fully grasp the concepts and workflow. This learning curve may hinder adoption in environments where multidisciplinary teams have limited prior experience with goal-oriented modeling or safety analysis techniques. Moreover, implementing RESafety in industrial or collaborative settings requires coordination among diverse stakeholders, including system engineers, safety analysts, and domain experts.

  Aligning their perspectives and maintaining model consistency across iterations can be challenging without adequate tool support. Tool limitations, such as the absence of fully integrated environments to handle both iStar4Safety and STPA artifacts, also restrict scalability and traceability in practice. Additionally, organizations must consider process integration aspects—such as how RESafety aligns with existing development standards and safety assurance workflows (e.g., ISO 26262 or IEC 61508)—to ensure its practical applicability. Therefore, while RESafety provides methodological rigor and traceability benefits, its successful adoption in real development contexts depends on addressing challenges related to training, tool support, process alignment, and the integration of interdisciplinary expertise.

- **Limited number of iterations in the illustrative application.** Our illustration of the RESafety process was limited to a single iteration and did not explore the results of applying multiple iterative cycles.

- **Ad-hoc identification of loss scenarios.** The identification of loss scenarios was conducted in an ad-hoc manner, as originally suggested by the STPA handbook. Although a more systematic and recently published approach by the technique's author is available, we opted for the traditional method, which may have limited the number of loss scenarios identified.

- **Low statistical power in the evaluation by safety experts.** The evaluation showed low statistical power in the subgroup of safety engineering professionals, due to limited participation from experts in that area.

6.4 Future Research

Based on the limitations identified in this work, several opportunities for future research can be outlined to strengthen and extend the RESafety process:

- **Development of a dedicated support tool.** One of the most pressing next steps is the development of a tool that enables the complete and semi-automated application of the RESafety process. A web-based prototype is currently under development and will be aligned with the final version of RESafety.
- **Provision of usage guidance.** The creation of a comprehensive usage guide is recommended, including common modeling mistakes and corresponding mitigation strategies, to assist new users and enhance the clarity and reliability of the process.
- **Formal case study in a controlled environment.** Conducting a formal case study in a safety-critical system, within a controlled environment, would enable a deeper investigation of the practical benefits of RESafety. Such a case study should evaluate aspects including (i) the scalability and completeness of the generated artifacts across iterations, (ii) the consistency and traceability between safety and requirements models, and (iii) the effort and time required for analysts to apply the process.
- **Execution of additional empirical studies.** Future research should include further empirical evaluations comparing RESafety with other safety-oriented approaches that address a broader range of aspects identified in the comparative analysis table presented in Chapter 2. Instead of focusing solely on approaches that produce similar artifacts, future comparisons should consider those that encompass a wider spectrum of integration, traceability, and modeling criteria, thereby providing a more comprehensive assessment of the relative strengths and limitations of RESafety.
- **Broader and more diverse participant samples.** Increasing the diversity and size of the participant pool in future evaluations—particularly among safety engineering professionals—would strengthen the statistical power of the results and allow for more generalizable conclusions.
- **Investigation of user learning curve and prerequisites.** Since RESafety relies on prior knowledge of iStar4Safety and STPA, subsequent studies may focus on simplifying the learning curve, as well as designing tailored training materials and onboarding strategies to facilitate adoption.
- **Exploration of maturity levels in RESafety.** Future investigations may explore the incorporation of maturity levels into the process, enabling organizations to adopt RESafety incrementally according to their capabilities and accumulated experience.

- **Connection with certification processes.** Examining how RESafety aligns with formal certification standards for safety-critical systems (e.g., IEC 61508, ISO 26262, or DO-178C) could enhance its applicability in regulated domains.
- **Refinement of Loss Scenario identification.** As Step 6.2 — Identify Potential Loss Scenarios — is considered one of the most complex for STPA users, further research is recommended to improve its clarity and reproducibility, potentially by integrating systematic heuristics and taxonomy-based identification techniques recently proposed within the STPA community.

Collectively, these research directions represent promising avenues to consolidate RESafety as a comprehensive and practically applicable process for the early modeling and traceability of safety requirements in safety-critical systems.

Acknowledgments

The authors would like to thank FACEPE (Fundação de Amparo à Ciência e Tecnologia de Pernambuco) for the financial support provided for the completion of this doctoral research (IBPG-0376-1.03/19).

References

(2012). MIL-STD-882e System Safety. Standard MIL-STD-882E, Department of Defense.

Adriaensen, A., Pintelon, L., Costantino, F., Gravio, G. D., & Patriarca, R. (2021). An stpa safety analysis case study of a collaborative robot application. *IFAC-PapersOnLine*, 54(1):534–539. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021.

Albuquerque, D., Castro, J., Ribeiro, S., & Heineck, T. (2017). Requirements engineering for robotic system: A systematic mapping study.

Alturayeif, N. & Hassine, J. (2025). Refactoring goal-oriented models: a linguistic improvement using large language models. *Software and Systems Modeling*.

Anton, A. I. (1996). Goal-based requirements analysis. In *Proceedings of the 2Nd International Conference on Requirements Engineering (ICRE '96)*, 136–.

Asnar, Y., Giorgini, P., & Mylopoulos, J. (2011). Goal-driven risk assessment in requirements engineering. *Requir. Eng.*, 16(2):101–116.

Bas, E. (2020). Stpa methodology in a socio-technical system of monitoring and tracking diabetes mellitus. *Applied Ergonomics*, 89:103190.

Berry, D. M. (1998). The safety requirements engineering dilemma. In *Proceedings of the 9th International Workshop on Software Specification and Design*, 147–.

Broomfield, E. & Chung, P. (1997). Safety assessment and the software requirements specification. *Reliability Engineering System Safety*, 55(3):295 – 309.

Börstler, J., bin Ali, N., Petersen, K., & Engström, E. (2024). Acceptance behavior theories and models in software engineering — a mapping study. *Information and Software Technology*, 172:107469.

da Silva, L. F. & Oliveira, E. (2020). Evaluating usefulness, ease of use and usability of an uml-based software product line tool. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, 798–807.

Dalpiaz, F., Franch, X., & Horkoff, J. (2016). istar 2.0 language guide. *CoRR*, abs/1605.07767.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). *Selecting Empirical Methods for Software Engineering Research*, 285–311. Springer London, London.

Firesmith, D. (2004). Engineering safety requirements, safety constraints, and safety-critical requirements. *Journal of object technology*, 3(3):27–27.

Glass, R. L. (1994). The software-research crisis. *IEEE Softw.*, 11(6):42–47.

Gonzalez Atienza, M. A., Vanoost, D., Kleihorst, R., & Pissoort, D. (2024). System-theoretic process analysis approach to analyse emi-related hazards and prioritise loss scenarios. *Safe AI Systems*, 95–114.

Gorschek, T., Garre, P., Larsson, S., & Wohlin, C. (2006). A model for technology transfer in practice. *IEEE Software*, 23(6):88–95.

Horkoff, J. & Yu, E. (2016). Interactive goal model analysis for early requirements engineering. *Requir. Eng.*, 21(1):29–61.

Hospital das Clínicas da UFPE (2023). 2nd innovation showcase at hc-ufpe highlights robotics and ai applied to medication dispensing. Originally titled in Portuguese: "2º Showcase de Inovação do HC-UFPE destaca robótica e IA aplicadas na dispensação de medicamentos". Accessed: 2025-07-25.

Houhamdi, Z., Athamena, M. R., Athamena, B., & ElRefae, G. (2024). Decision support in goal-oriented requirements engineering. In *2024 25th International Arab Conference on Information Technology (ACIT)*, 1–4.

Kitchenham, B. A. & Pfleeger, S. L. (2008). *Personal Opinion Surveys*, 63–92. Springer London, London.

Knight, J. C. (2002). Safety critical systems: Challenges and directions. In *Proceedings of the 24th International Conference on Software Engineering*, 547–550.

Lai, P. C. (2017). The literature review of technology adoption models and theories for the novelty technology. *Journal of Information Systems and Technology Management*, 14(1):21–38.

Lapouchnian, A. (2005). Goal-oriented requirements engineering: An overview of the current research.

Leveson, N. G. (1995). *Safeware: System Safety and Computers*. ACM, New York, NY, USA.

Leveson, N. G. (2011). *Engineering a Safer World: Systems Thinking Applied to Safety*. Mit Press, Massachusetts, London, England.

Leveson, N. G. & Thomas, J. P. (2018). *STPA Handbook*. first edition.

Lutz, R. R. (1993). Targeting safety-related errors during software requirements analysis. *SIGSOFT Softw. Eng. Notes*, 18(5):99–106.

Lutz, R. R. (2000). Software engineering for safety: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, 213–226.

Manjunath, M., Jesus Raja, J., & Daun, M. (2025). Early model-based safety analysis for collaborative robotic systems. *IEEE Transactions on Automation Science and Engineering*, 22:17523–17534.

Martinazzo, A. (2022). Gerenciamento de risco de uma bomba de infusão de insulina de baixo custo. Master's thesis, Universidade Federal de São Paulo (UNIFESP). In English: Risk management of a low-cost insulin infusion pump.

Martins, L. E. G., Faria, H. d., Vecchete, L., Cunha, T., Oliveira, T. d., Casarini, D. E., & Colucci, J. A. (2015). Development of a low-cost insulin infusion pump: Lessons learned from an industry case. In *Proceedings of the 2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, 338–343.

Martins, L. E. G. & Gorschek, T. (2017). Requirements engineering for safety-critical systems: Overview and challenges. *IEEE Software*, 34(4):49–57.

Martins, L. E. G. & Gorschek, T. (2021). *Requirements Engineering for Safety-Critical Systems*. River, Denmark.

McDermid, J., Nicholson, M., Pumfrey, D., & Fenelon, P. (1995). Experience with the application of hazop to computer-based systems. In *COMPASS '95 Proceedings of the Tenth Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security'*, 37–48.

Morandini, M., Penserini, L., Perini, A., & Marchetto, A. (2017). Engineering requirements for adaptive systems. *Requir. Eng.*, 22(1):77–103.

Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, 35–46.

Opdahl, A. L. & Sindre, G. (2009). Experimental comparison of attack trees and misuse cases for security threat identification. *Information and Software Technology*, 51(5):916–932. SPECIAL ISSUE: Model-Driven Development for Secure Information Systems.

Peraldi-Frati, G., Boniol, F., Fabre, J.-C., & Waeselynck, H. (2019). Comparison of the fmea and stpa safety analysis methods–a case study. *Software Quality Journal*, 27(1):349–387.

Pimentel, J., de Castro, J. B., Ribeiro, S. M., Ramos, A. S., & Ramos, B. (2019). Creating modelling tools for i* language extensions. In *iStar@ER*.

Popping, R. (2015). Analyzing open-ended questions by means of text analysis procedures. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, 128(1):23–39.

Ribeiro, M. (2019). Desenvolvimento de uma extensão da linguagem de modelagem istar para sistemas críticos de segurança - istar4safety. Master's thesis, Universidade Federal de Recife.

Ribeiro, M., Castro, J., & Argenton, R. (2024). Integrating stpa with safety requirements modeling. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software*, 561–567.

Ribeiro, M., Castro, J., & Pimentel, J. (2019a). istar for safety-critical systems. In Pimentel, J., Carvallo, J. P., & López, L., editors, *Proceedings of the 12th International i* Workshop co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4th, 2019*, 2490.

Ribeiro, M., Castro, J., & Pimentel, J. (2019b). istar for safety-critical systems. In Pimentel, J., Carvallo, J. P., & López, L., editors, *Proceedings of the 12th International i* Workshop co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4th, 2019*, 2490.

Ribeiro, M., Castro, J., Ramos, R., Lencastre, M., & Santos, A. (2023a). Requirements4safety – construindo uma técnica para modelagem de requisitos iniciais de segurança. In *Proceedings of the 26th Workshop on Requirements Engineering (WER 2023)*. Originally published in Portuguese.

Ribeiro, M., Castro, J., Ramos, R. A., Lencastre, M., Santos, A., & Pastor, O. (2023b). Integrating goal oriented requirements modeling and safety analysis with requirements4safety. In Liaskos, S., Portugal, R. L. Q., & Maté, A., editors, *Proceedings of the 16th International iStar Workshop (iStar 2023) co-located with 31st IEEE International Requirements Engineering 2023 Conference (RE 2023), Hannover, Germany, September 03-04, 2023*, 3533:40–46.

Ribeiro, M., Castro, J., Vilela, J., & Pimentel, J. (2019c). istar4safety: Uma extensão de istar para modelagem de requisitos de segurança em sistemas críticos. In Lencastre, M., Ridao, M., & de Sá Sousa, H. P., editors, *Anais do WER19 - Workshop em Engenharia de Requisitos, Recife, Brasil, August 13-16, 2019*.

Ruijters, E. & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15-16:29–62.

Scholz, S. & Thramboulidis, K. (2013). Integration of model-based engineering with system safety analysis. *International Journal of Industrial and Systems Engineering*, 15(2):193–215.

Sharifi, S., Amyot, D., Mylopoulos, J., McLaughlin, P., & Feodoroff, R. (2022). Towards improved certification of complex fintech systems – a requirements-based approach. In *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 205–214.

Sharifi, S., McLaughlin, P., Amyot, D., & Mylopoulos, J. (2020). Goal modeling for fintech certification. In Guizzardi, R. S. S. & Mussbacher, G., editors, *Proceedings of the Thirteenth International iStar Workshop co-located with 28th IEEE International Requirements Engineering Conference (RE 2020)*, 2641:73–78.

Sheikh Bahaei, S. & Gallina, B. (2024). Assessing risk of ar and organizational changes factors in socio-technical robotic manufacturing. *Robotics and Computer-Integrated Manufacturing*, 88:102731.

Soares, M. S. & do Nascimento, R. P. C. (2014). Evaluation of sysml diagrams to document requirements using tam. In *Proceedings of the 7th Euro American Conference on Telematics and Information Systems*.

Sommerville, I. (2010). *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition.

Stålhane, T. & Myklebust, T. (2016). Early safety analysis. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, 19:1–19:6.

Van Lamsweerde, A. & Letier, E. (2000). Handling obstacles in goal-oriented requirements engineering. *IEEE Trans. Softw. Eng.*, 26(10):978–1005.

Vilela, J., Castro, J., Martins, L. E. G., & Gorschek, T. (2017a). Integration between requirements engineering and safety analysis. *J. Syst. Softw.*, 125(C):68–92.

Vilela, J., Castro, J., Martins, L. E. G., & Gorschek, T. (2018). Safe-re: A safety requirements metamodel based on industry safety standards. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, 196–201.

Vilela, J., Castro, J., Martins, L. E. G., Gorschek, T., & Silva, C. (2017b). Specifying safety requirements with gore languages. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, 154–163.

Vilela, J., Castro, J., Martins, L. E. G., Gorschek, T., & Silva, C. (2017c). Specifying safety requirements with gore languages. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, 154–163.

Vilela, J., Silva, C., Castro, J., Martins, L. E. G., & Gorschek, T. (2019). Sarssi*: a safety requirements specification method based on stamp/stpa and i* language. In *Anais do I Brazilian Workshop on Large-scale Critical Systems*, 17–24.

Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X., Makris, S., & Chryssolouris, G. (2019). Symbiotic human-robot collaborative assembly. *CIRP Annals*, 68(2):701–726.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., & Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

Yu, E. S.-K. (1995). *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, Toronto, Ont., Canada, Canada. AAINN02887.

Yu, E. S. K. (1997). Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, 226–.

Zhang, Y., Jetley, R., Jones, P. L., & Ray, A. (2011). Generic safety requirements for developing safe insulin pump software. *Journal of Diabetes Science and Technology*, 5(6):1403–1419. PMID: 22226258.

Zhang, Y., Jones, P. L., & Jetley, R. (2010). A hazard analysis for a generic insulin infusion pump. *Journal of Diabetes Science and Technology*, 4(2):263–283. PMID: 20307387.

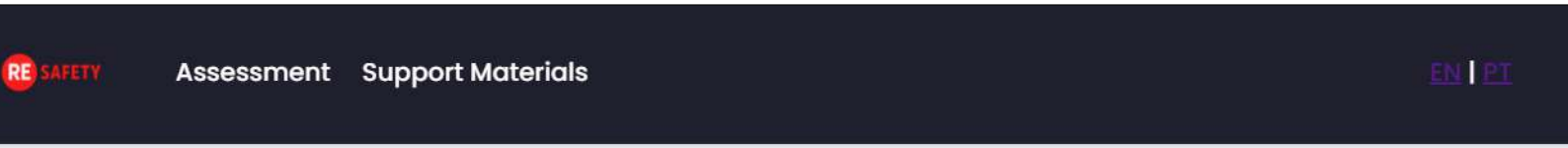
Zoughbi, G., Briand, L., & Labiche, Y. (2011). Modeling safety and airworthiness (rtca do-178b) information: Conceptual model and uml profile. *Softw. Syst. Model.*, 10(3):337–367.

A

RESafety Evaluation Support Website

A support website was developed to guide participants through the evaluation of the RESafety process. The site included the complete set of materials required to perform the evaluation, such as tutorial videos, supporting documents, and access to the questionnaire.

The questionnaire itself was developed using the Google Forms platform and made available to participants after they had completed the training phase involving the tutorial videos and analysis of the provided artifacts.

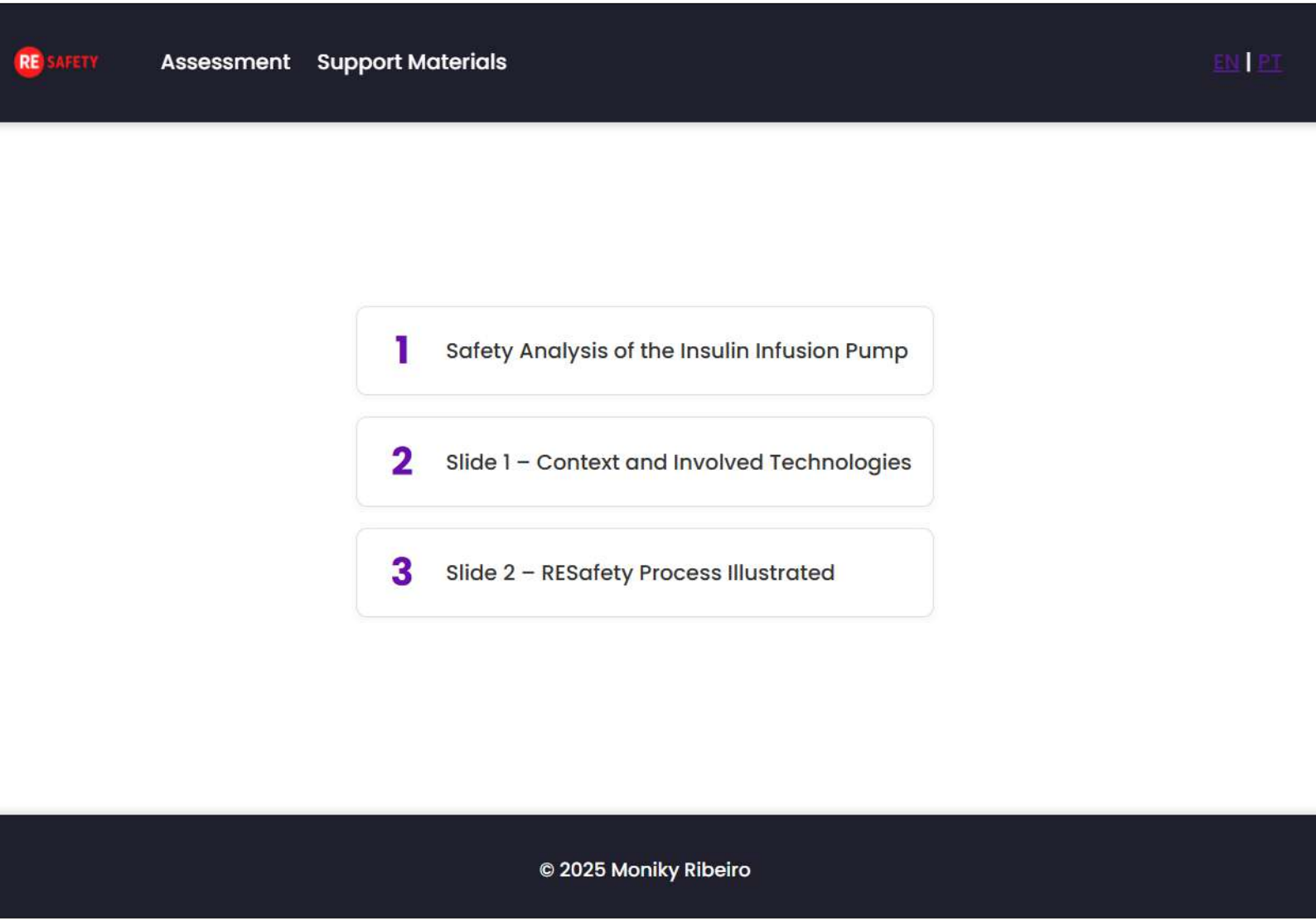


Dear evaluator,

Please follow the indicated order: watch tutorial videos 1 to 3 and then complete the questionnaire in step 4. The "**Support Materials**" tab contains the safety analysis artifact used in the example, as well as the presentation PDFs.

We sincerely appreciate your participation — it is extremely important for evaluating our process.

1. Video 1
   Introduction and Technology Overview
2. Video 2
   RESafety Process and Example - Step 1 to 2
3. Video 3
   RESafety Process and Example - Step 3 to 7
4. Questionnaire


RE SAFETY
Assessment
Support Materials
EN | PT
1 Safety Analysis of the Insulin Infusion Pump
2 Slide 1 – Context and Involved Technologies
3 Slide 2 – RESafety Process Illustrated
© 2025 Moniky Ribeiro

B

TAM-Based Questionnaire Used in the Evaluation

This appendix presents the questionnaire used to evaluate the RESafety process, based on the Technology Acceptance Model (TAM). The questionnaire was designed to assess constructs such as perceived usefulness (PU) and perceived ease of use (PEOU), and was implemented as a self-administered form using the Google Forms platform.

Participants responded to the questionnaire after completing the training phase, which included watching tutorial videos and analyzing RESafety artifacts. The full version of the questionnaire is shown below.

Informed Consent and Personal Profile

Please provide your email address so we can contact you in the future and share the results of the evaluation process.

* Indicates required question

1. E-mail *

Untitled Section

Dear participant,

We invite you to take part in the evaluation of the **RESafety process**, a process for modeling safety requirements, through a series of short questionnaires.

This evaluation seeks to gather expert feedback on the **steps and artifacts** produced by the RESafety process.

We would like to emphasize that:

1. Your participation is entirely voluntary.
2. We are solely interested in your professional opinion regarding the research topic and its outcomes.
3. You may withdraw your participation and revoke your consent at any time.
4. The data collected will be used in a way that ensures the anonymity of all respondents.

Researchers involved:

• Moniky Ribeiro (Doctoral Student, Postgraduate Program in Computer Science – Cin/UFPE) – smsr@cin.ufpe.br

• Prof. Dr. Jaelson Castro (Cin/UFPE) – jbc@cin.ufpe.br

• Prof. Dr. Ricardo Argenton Ramos (UNIVASF) – ricardo.aramos@univasf.edu.br

2. I agree to the terms: *

Check all that apply.

☐ Yes

Personal Profile

We appreciate your contribution. The following questions aim to gather your insights regarding safety and system requirements practices.

3. Gender *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Other

4. Age Range *

Mark only one oval.

- ◯ 18–29 years
- ◯ 30–44 years
- ◯ 45–59 years
- ◯ 60 years or older

5. Highest Level of Education *

Mark only one oval.

- ◯ Undergraduate Degree
- ◯ Lato Sensu Postgraduate (Specialization)
- ◯ Master's Degree (Stricto Sensu)
- ◯ Doctorate / Ph.D. (Stricto Sensu)
- ◯ Postdoctoral Research / Fellowship
- ◯ Other: _____

6. Country or Region of Residence *

Mark only one oval.

- ◯ Brazil
- ◯ Other Country
- ◯ Prefer not to say

7. What is your primary professional or research area in the field of safety and/or requirements engineering? *

Mark only one oval.

- ◯ Requirements Engineering
- ◯ Safety Engineering

8. Years of Professional Experience in Primary Area *

Mark only one oval.

- ◯ <1
- ◯ 1-2
- ◯ 3-5
- ◯ 6-10
- ◯ >10
- ◯ I have no professional experience in this area

RESafety Questions

Please indicate how much you agree with each of the following statements regarding the RESafety process. Use a scale from 1 to 7, where 1 means "Strongly Disagree" and 7 means "Strongly Agree."

9. Q01 - RESafety enables me to accomplish safety analysis tasks more quickly *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

10. Q02 - Using RESafety improves my safety analysis *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

11. Q03 - Using RESafety increases my productivity. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

12. Q04 - Using RESafety enhances my effectiveness in performing safety analysis. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

13. Q05 - Using RESafety makes it easier to do my safety analysis. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

14. Q06 - Overall, I find RESafety useful for safety analysis of Safety Critical Systems. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

15. Q07 - The use of RESafety does not require a lot of my mental effort. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

16. Q08 - I find it easy to follow the steps of RESafety. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

17. Q09 - The steps of RESafety are clear and understsndable. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

18. Q10 - The steps of RESafety are flexible. *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

19. Q11 - It is easy to become skillful in using RESafety *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

20. Q12 - I find the RESafety process easy to use *

Mark only one oval.

	1	2	3	4	5	6	7	
Stro	◯	◯	◯	◯	◯	◯	◯	Strongly Agree

21. Do you have any additional remarks? Please write them below. Your opinion is very important to our research.

22. Do you allow us to contact you to clarify any questions? *

Mark only one oval.

- Yes
- No

23. Please leave your contact information (preferably email and/or WhatsApp). Your participation will remain anonymous.

C

Tutorial 1 – RESafety Context and Involved Technologies

This appendix presents the first slide used in the tutorial material provided to participants during the evaluation of the RESafety process. The slide introduces the context and the main technologies involved in the process.

RESafety

Requirements Engineering for Safety

PhD Student: Moniky Ribeiro
Advisor: Prof. Jaelson Castro
Co-advisor: Prof. Ricardo Argenton

Context

- Developing SCS involves some additional activities not common in non-critical contexts
  - Define high-level safety requirements
  - Identify accidents and hazards
  - Investigate if the combined behavior of system's components and stakeholders may lead to hazards
  - Refine safety requirements towards system's design
  - Assure system's safety
- Hazard elimination or mitigation impacts system's requirements:
  - It may derive new functional requirements
  - It may derive new non-functional requirements
  - It may derive new architectural decisions
  - Or it may impact existing ones

Context

A Safety-Critical System (SCS) is defined as:

A system which, if it fails or behaves unexpectedly, can cause accidents resulting in injury or death, damage to property or the environment, or significant financial loss (Leveson, 1995; Leveson, 2011).

RESafety

IStar4Safety
- Safety goals
- Hazards
- Mitigation strategies (Safety Tasks/ Safety Resources)

STPA
- Accidents
- Hazards
- Constraints
- Unsafe Control Actions
- Loss Scenarios

RESafety

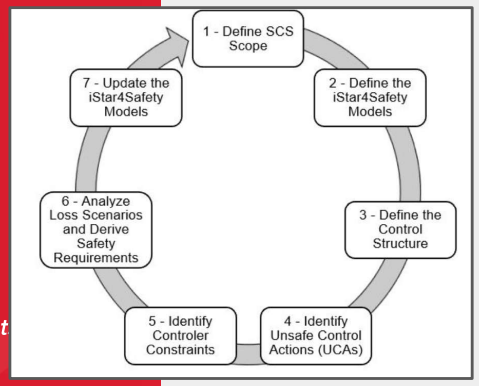
Requirements Engineering
-GORE Models

Safety Engineering
Safety Analysis Techniques

RESafety

The 7-Step Process

- *Iterative and incremental*
- *Relies on iStar4Safety models and STPA element*

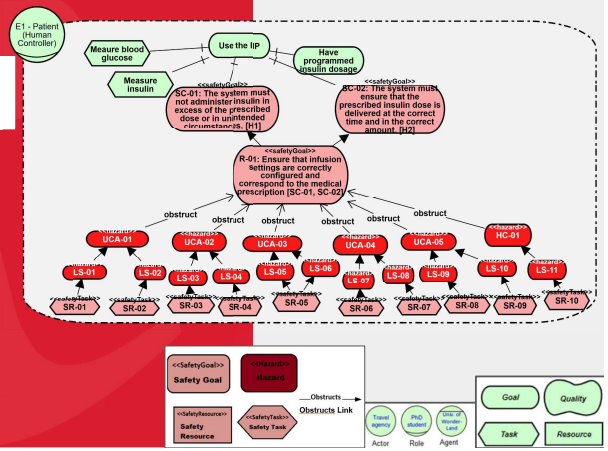


iStar4Safety

Early Safety Requirements Modeling

Overview

(Ribeiro, 2019)

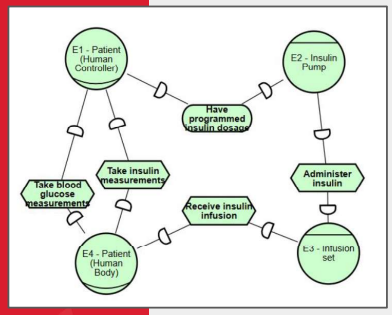


iStar 2.0

Early requirements Modeling

Overview

(Yu, 1995) and (DALPIAZ; FRANCH; HORKOFF, 2016)

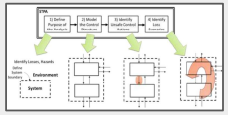


STPA

System-Theoretic Process Analysis

(Leveson, 2011, Leveson & Thomas, 2018)

- Developed by Nancy Leveson (MIT).
- A method for hazard analysis in complex socio-technical systems.
- Based on Systems Theory, not failure-based models like FMEA or FTA.

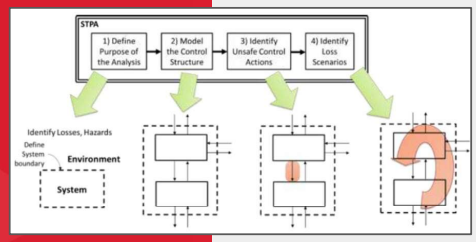


STPA

System-Theoretic Process Analysis

(Leveson, 2011, Leveson & Thomas, 2018)

- Definition of the purpose of the analysis
- Control structure modeling
- Identification of UCAs - Unsafe Control Actions
- Identification of Loss Scenarios



References

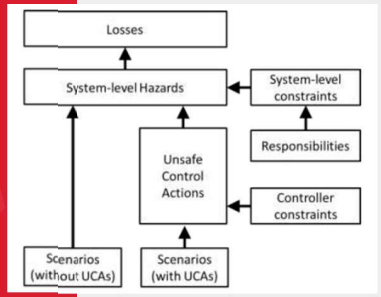
1. Leveson, N. G. (2011). Engineering a Safer World: Systems Thinking Applied to Safety. Mit Press, Massachusetts, London, England.
2. Leveson, N. G. & Thomas, J. P. (2018). STPA Handbook. first edition.
3. Ribeiro, M. (2019). Desenvolvimento de uma extensão da linguagem de modelagem istar para sistemas críticos de segurança - istar4safety. Master's thesis, Universidade Federal de Recife.
4. Yu, E. S.-K. (1995). Modelling Strategic Relationships for Process Reengineering. PhD thesis, Toronto, Ont., Canada, Canada. AAINN02887.

STPA

System-Theoretic Process Analysis

(Leveson, 2011, Leveson & Thomas, 2018)

Outputs and Traceability in STPA



Leveson & Thomas, 2018.

Thanks!

Contacts:

Moniky Ribeiro-> monikyr@gmail.com **or** smsr@cin.ufpe.br

Jaelson Castro-> jbc@cin.ufpe.br


50
anos
Centro de
Informática
UFPE

D

Tutorial 2 – RESafety Process and Application

This appendix presents the second slide used in the tutorial material provided to participants during the evaluation of the RESafety process. The slide outlines the steps of the RESafety process and illustrates them through the modeling of the Insulin Infusion Pump System (IIPS).

RESafety

Requirements Engineering for Safety

PhD Student: Moniky Ribeiro
Advisor: Prof. Jaelson Castro
Co-advisor: Prof. Ricardo Argenton

RESafety

Insulin Infusion Pump (IIP) Example

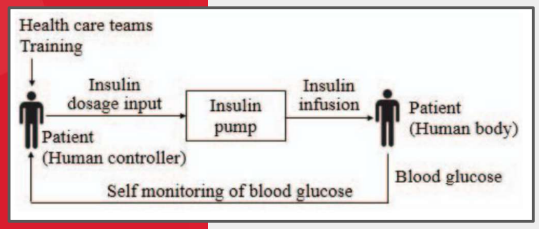
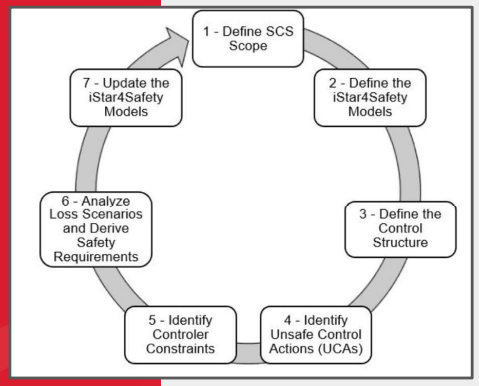


Figure 1 – Overview of the Insulin Infusion Pump System [1]

RESafety

The 7-Step Process



Insulin Infusion Pump (IIP)

Types of Insulin Delivery

- Basal: A constant, low-level infusion throughout the day and night. Example: Up to five programmable basal profiles over 24h.
- Bolus: A larger dose triggered by meals or to correct high blood sugar. Delivered manually or based on user programming.

Insulin Infusion Pump (IIP)

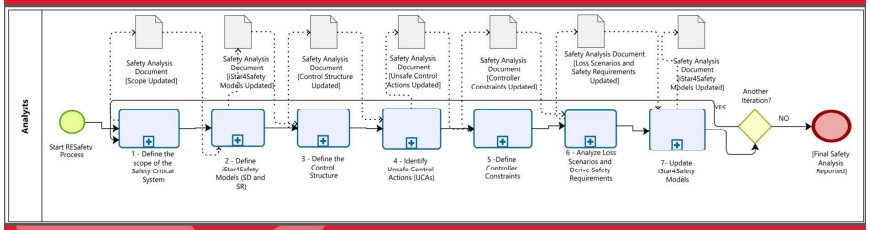
Components of a Typical Pump

- User interface: LCD screen and audio alarms.
- Hardware parts: Microprocessor, battery, infusion mechanism, insulin reservoir, and catheter.

Figure 2 – Retrieved from [2]

RESafety

BPMN Diagram of the Process

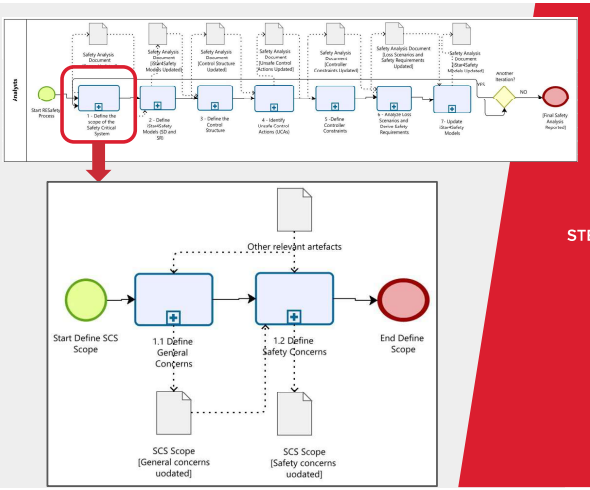


Link to the IIP Example Analysis:
https://docs.google.com/document/d/1ubwn6wNi_9WhdnSAAOxyeq6hOL1R7Ldqu0FSG1769yc/edit?tab=t.0

Insulin Infusion Pump (IIP)

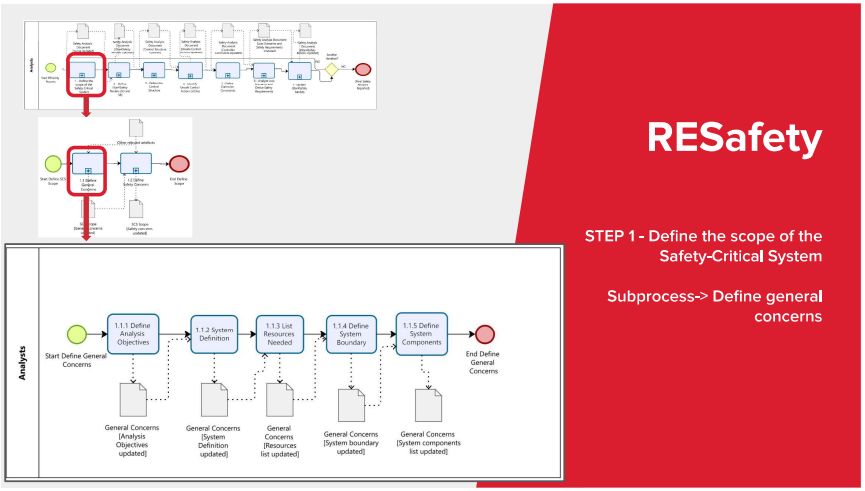
Safety implications

- Main accidents:
  - Overdose → Hypoglycemia (low blood sugar)
  - Underdose → Hyperglycemia (high blood sugar)
- Other accidents: skin infections, battery failure, or device malfunction, etc.



RESafety

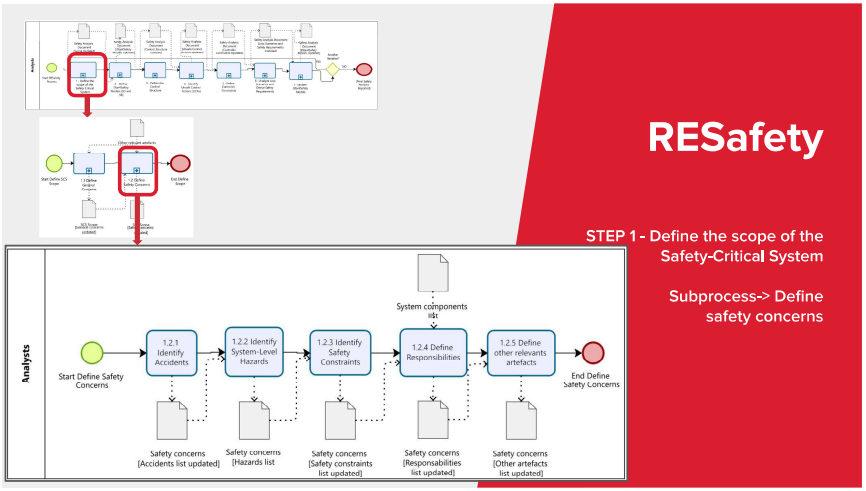
STEP 1 - Define the scope of the Safety-Critical System



RESafety

STEP 1 - Define the scope of the Safety-Critical System

Subprocess-> Define general concerns



RESafety

STEP 1 - Define the scope of the Safety-Critical System

Subprocess-> Define safety concerns

STEP 1 - Define the scope of the SCS

1.1 General Concerns

1.1.1 Analysis Objectives

The purpose of this analysis is to model an Insulin Infusion Pump (IIP) through the iterative RESafety process, generating successive refinements of the system's safety analysis artifacts.

1.1.2 System Definition

The Insulin Infusion Pump (IIP), a safety-critical system, is designed to support the treatment of Type 1 Diabetes Mellitus. Automated IIPs enhance treatment flexibility by managing multiple stages of insulin delivery, effectively mimicking physiological responses. These devices administer both rapid-acting (bolus) and continuous (basal) insulin doses.

1.1.3 Resources Needed for Analysis

- **Articles**: Martinazzo (2022); Martins et al. (2015); Zhang et al. (2011, 2010); Bas (2020); Gonzalez Atienza et al. (2024)
- **Books**: Leveson & Thomas (2018); Martins & Gorschek (2021)
- General Guidelines and Manuals

1.1.4 System Boundary

The system boundary encompasses activities from the moment the patient configures the infusion settings until the correct dosage is delivered via the catheter.

1.1.5 Components

- Patient
- Infusion Insulin Pump
- Infusion Set

STEP 1 - Define the scope of the SCS

1.2 Safety Concerns

1.2.1 Identify Accidents

- **A1 -** Risk of death
- **A2 -** Risk of injury

1.2.2 Identify System-Level Hazards

- **H1 -** Hypoglycemia [A1, A2]
- **H2 -** Hyperglycemia [A2]

1.2.3 Identify System Constraints

- **SC-01 -** The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
- **SC-02 -** The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]

1.2.4 Identify the responsibilities

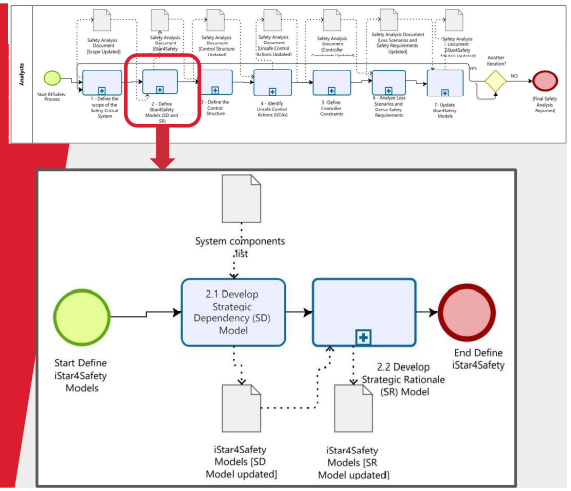
Entity	Responsability
E1 – Patient (Human Controller)	**R-01: Ensure that infusion settings are correctly configured and correspond to the medical prescription [SC-01, SC-02]** **R-02: Verify that the device interface confirms the programmed dose before administration [SC-01]**
E2 - Insulin Infusion Pump	...
E3 - Infusion Set	...
E4 - Patient (Human Body)	...

1.2.5 Other Artifacts

- *Not applicable*

RESafety

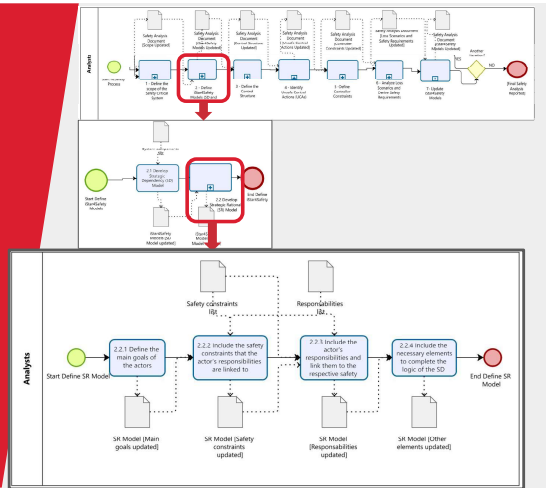
STEP 2 - Define iStar4Safety Models (SD and SR)



RESafety

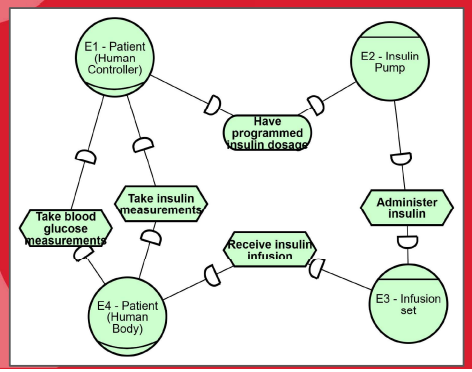
STEP 2 - Define iStar4Safety Models (SD and SR) -

Subprocess-> Develop Strategic Rationale (SR) model



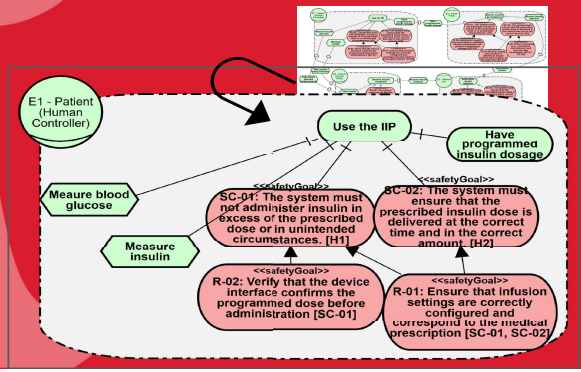
STEP 2 - Define iStar4Safety Models

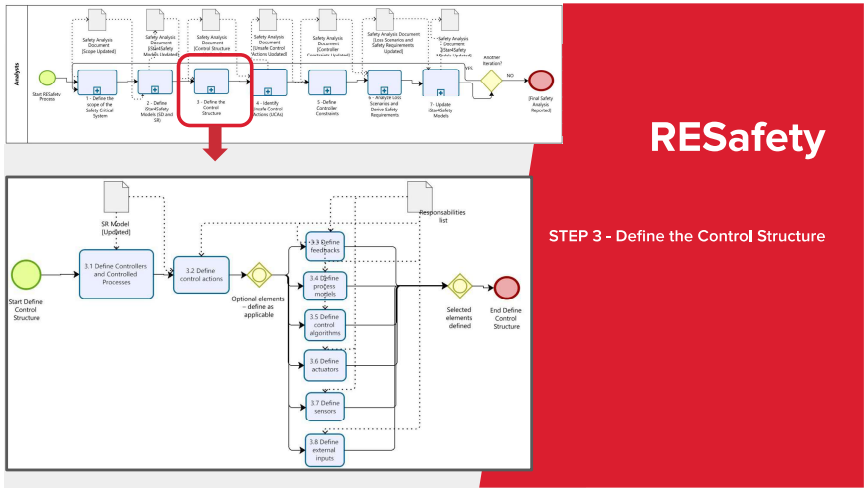
SD Model



STEP 2 - Define iStar4Safety Models

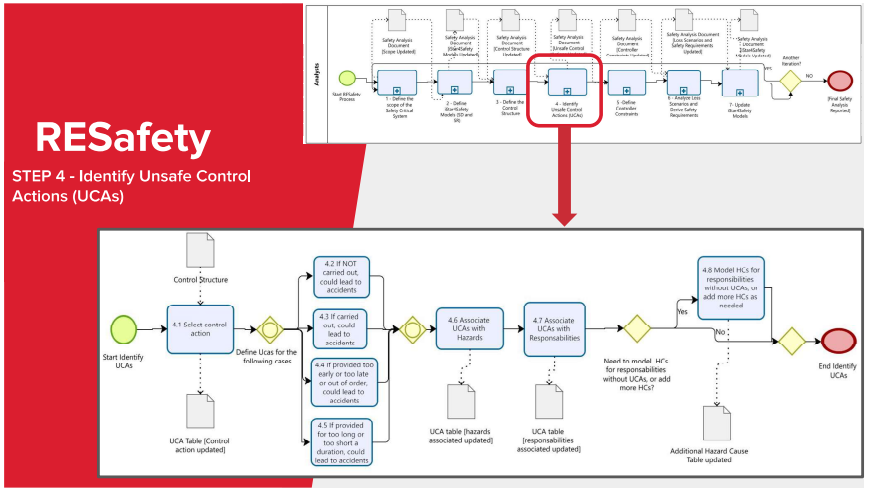
SD and SR Models





RESafety

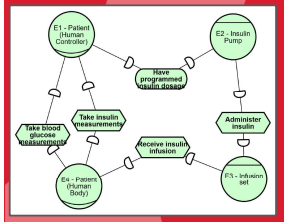
STEP 3 - Define the Control Structure

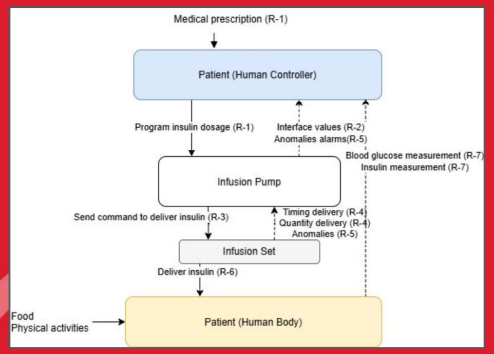


RESafety

STEP 4 - Identify Unsafe Control Actions (UCAs)

STEP 3 - Define the Control Structure

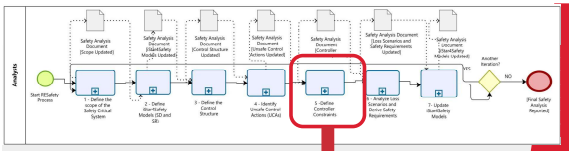


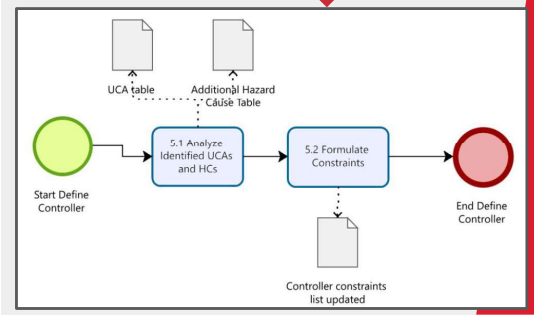


STEP 4 - Define UCAs

Control Action	From/To	Not Providing Causes Hazard	Providing Causes Hazard	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Program insulin dosage (R-1)	Patient / Infusion Pump	**UCA-01:** Patient does not provide "Program insulin dosage" when insulin is required, leading to underdose [H1]	**UCA-02:** Patient provides "Program insulin dosage" with a value higher than prescribed, leading to overdose [H2] **UCA-03:** Patient provides "Program insulin dosage" with a value lower than prescribed, leading to underdose [H1]	**UCA-04:** Patient provides "Program insulin dosage" too late, leading to hyperglycemia [H1] **UCA-05:** Patient provides "Program insulin dosage" too early, leading to premature insulin administration and resulting in hypoglycemia [H2]	*Not applicable*

Hazard Cause
HC-01: The pump is misplaced or inaccessible to the patient [H2]



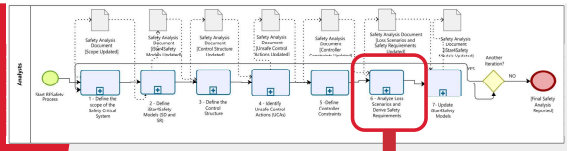


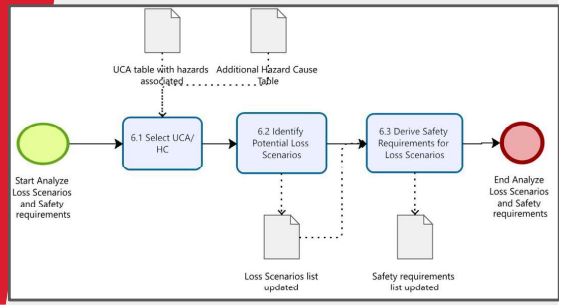
RESafety

STEP 5 - Define controller constraints

RESafety

STEP 6 - Analyze Loss Scenarios and derive safety requirements





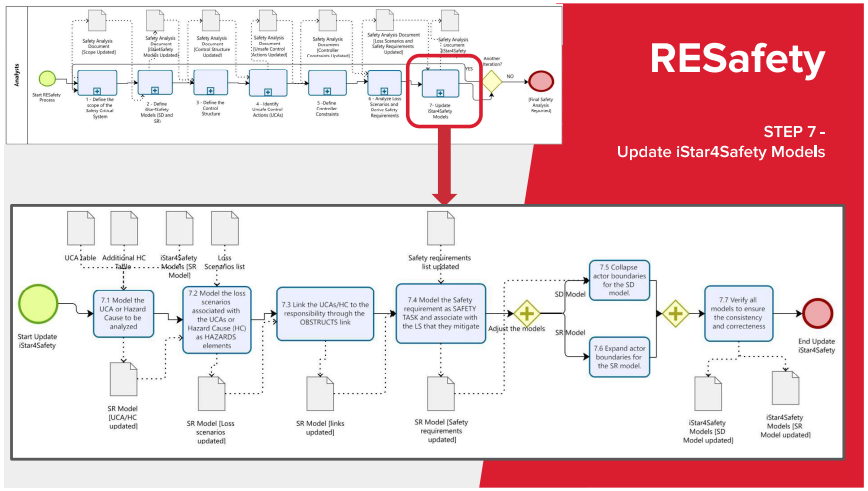
STEP 5 - Define Controller Constraints

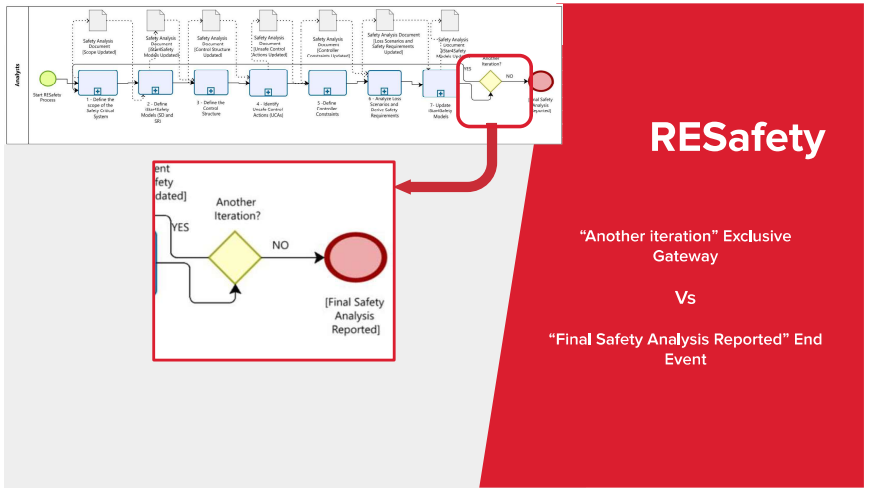
Unsafe Control Action	Controller Constraint
UCA-01: Patient does not provide "Program insulin dosage" when insulin is required, leading to underdose. [H1]	**C-01:** The patient must program the insulin dosage whenever insulin is required, according to clinical guidance. [UCA-01]
UCA-02: Patient provides "Program insulin dosage" with a value higher than prescribed, leading to overdose. [H2]	**C-02:** The patient must ensure the programmed insulin dosage does not exceed the value prescribed by the physician. [UCA-02]
UCA-03: Patient provides "Program insulin dosage" with a value lower than prescribed, leading to underdose. [H1]	**C-03:** The patient must verify that the programmed dosage meets the minimum prescribed threshold to avoid underdosing. [UCA-03]
UCA-04: Patient provides "Program insulin dosage" too late, leading to hyperglycemia. [H1]	**C-04**: The patient must program the insulin dosage in a timely manner, according to the prescribed administration window. [UCA-04]
UCA-05: Patient provides "Program insulin dosage" too early, leading to premature insulin administration and resulting in hypoglycemia. [H2]	**C-05:** The patient must not program the insulin dosage before the appropriate physiological or dietary conditions occur. [UCA-05]
HC-01: The pump is misplaced or inaccessible to the patient.	**C-06:** The insulin pump must always be correctly placed and readily accessible to the patient.

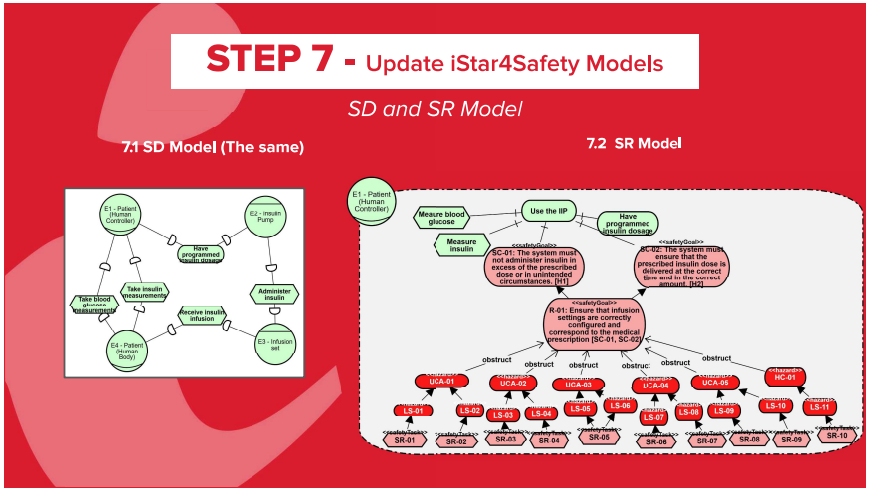
STEP 6 - Analyze Loss Scenarios and derive safety requirements

UCA	Loss Scenario (LS)	Safety Requirement (SR)
UCA-01: Patient does not provide "Program insulin dosage" when insulin is required, leading to underdose [H1]	**LS-01:** The patient forgets to program the dose after the meal, resulting in hyperglycemia. [UCA-01] *Martinazzo (2022)* **LS-02:** The system does not issue a reminder to program the dose after detecting a meal event. [UCA-01] *Ribeiro et al. (2024)*	**SR-01:** The system shall generate an alert if insulin is not programmed within 15 minutes after a meal is detected. [LS-01] *Zhang et al. (2011)* **SR-02:** The interface must maintain a visible warning if no insulin programming is detected post-meal. [LS-02] *Ribeiro et al. (2024)*
...	...	...

Hazard Cause	Loss Scenario	Safety requirement
HC-01: The pump is misplaced or inaccessible to the patient.	**LS-11**: The patient is in a critical condition and does not remember where the pump was placed.	**SR-10**: The pump must have an associated mobile application that allows a "locate pump" function to trigger an audible alarm when activated.







References

1. A. Martinazzo, L. E. G. Martins, S. V. Aredes and T. S. Cunha, "Risk Management of a Low-cost Insulin Infusion Pump: A Case Study with a Brazilian Company," 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS), Aveiro, Portugal, 2021.
2. WikEM. (n.d.). Insulin infusion device complication. Retrieved June 3, 2025, from http://medbox.iiab.me/modules/en-wikem/wiki/Insulin_infusion_device_complication.html

Thanks!

Contacts:

Moniky Ribeiro-> monikyr@gmail.com **or**
smsr@cin.ufpe.br

Jaelson Castro-> jbc@cin.ufpe.br

E

Analysis Insulin Infusion Pump System

This appendix presents the safety analysis performed using the RESafety process for an insulin infusion pump system, which served as the basis for evaluating the process. This analysis was used as an example in the tutorial videos and slides shown to the survey participants. It is important to emphasize that the objective was to illustrate the application of the RESafety process. Therefore, the intention was not to deliver a comprehensive safety analysis of the system, but rather to support participants in understanding the process by modeling the functionalities of a portion of a safety-critical system.

RESafety Analysis
Insulin Infusion System

Author: Moniky Ribeiro
Date: June 2025
Project or Institute: CIn/UFPE
System: Insulin Infusion Pump (IIP)
Iteration: 1ª

Step 1 - Define Safety-Critical System (SCS) Scope

1.1. General Concerns

1.1.1 Analysis Objectives

The purpose of this analysis is to model an Insulin Infusion Pump (IIP) through the iterative RESafety process, generating successive refinements of the system's safety analysis artifacts.

1.1.2 System Definition

The Insulin Infusion Pump (IIP), a safety-critical system, is designed to support the treatment of Type 1 Diabetes Mellitus. Automated IIPs enhance treatment flexibility by managing multiple stages of insulin delivery, effectively mimicking physiological responses. These devices administer both rapid-acting (bolus) and continuous (basal) insulin doses.

1.1.3 Resources Needed for Analysis

- Articles:
    - Martinazzo (2022);
    - Martins et al. (2015);
    - Zhang et al. (2011, 2010);
    - Bas (2020);
    - Gonzalez Atienza et al. (2024)
- Books
    - Leveson & Thomas (2018);
    - Martins & Gorschek (2021)
- General Guidelines and Manuals

1.1.4 System Boundary

The system boundary encompasses activities from the moment the patient configures the infusion settings until the correct dosage is delivered via the catheter.

1.1.5 Components

- Patient
- Infusion Insulin Pump
- Infusion Set

1.2. Safety Concerns

1.2.1 Identify Accidents

- **A1** - Risk of death
- **A2** - Risk of injury

1.2.2 Identify System-Level Hazards

- **H1 -** Hypoglycemia [A1, A2]
- **H2 -** Hyperglycemia [A2]

1.2.3 Identify System Constraints

- **SC-01 -** The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
- **SC-02 -** The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]

1.2.4 Define the responsibilities

Entity	Responsability
E1 – Patient (Human Controller)	**R-01:** Ensure that infusion settings are correctly configured and correspond to the medical prescription [SC-01, SC-02] **R-02:** Verify that the device interface confirms the programmed dose before administration [SC-01]
E2 - Insulin Infusion Pump	**R-03:** Administer insulin only according to validated infusion parameters and prevent unauthorized dosages [SC-01] **R-04:** Monitor timing and quantity of delivery to ensure correct dose is given at the right time [SC-02] **R-05:** Detect anomalies (e.g., occlusions, over-delivery) and alert the user immediately [SC-01, SC-02]

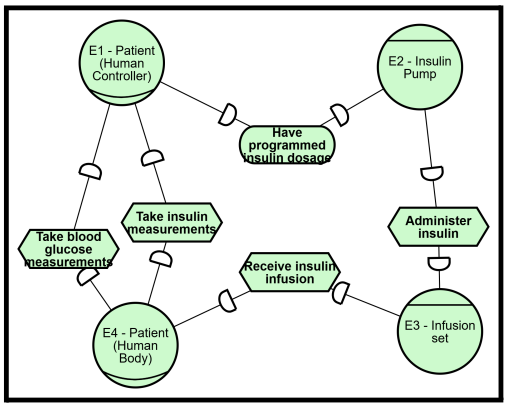
E3 - Infusion Set	**R-06:** Maintain physical integrity to prevent leaks or unintended flow of insulin [SC-01] **R-07:** Ensure correct and timely delivery of insulin from pump to patient [SC-02]
E4 - Patient (Human Body)	**R-08:** Respond physiologically to insulin in a way that is consistent with treatment expectations (acknowledging variability) [SC-02]

1.2.5. Other Artifacts

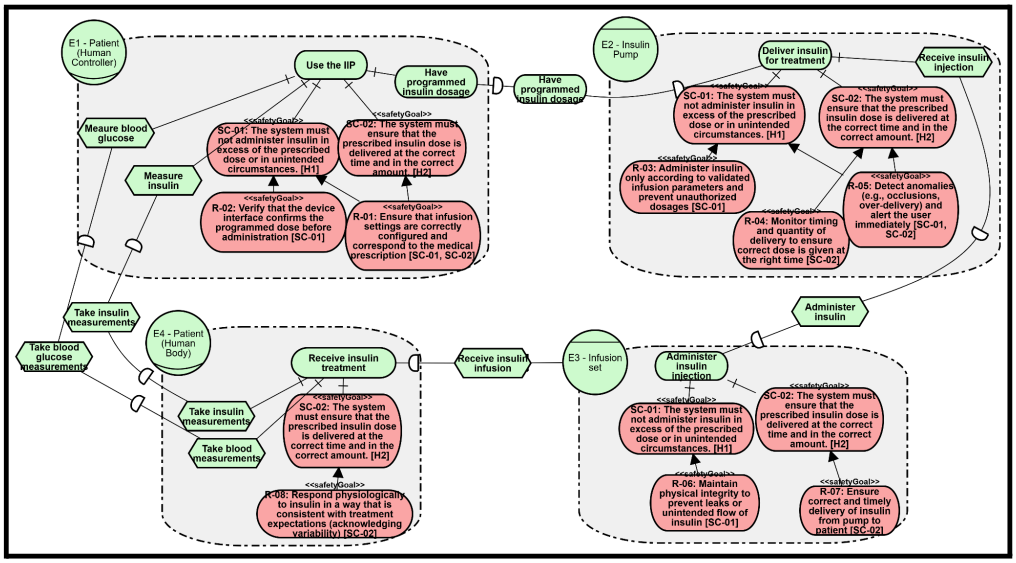
- *Not applicable*

Step 2 - Define the iStar4Safety Models

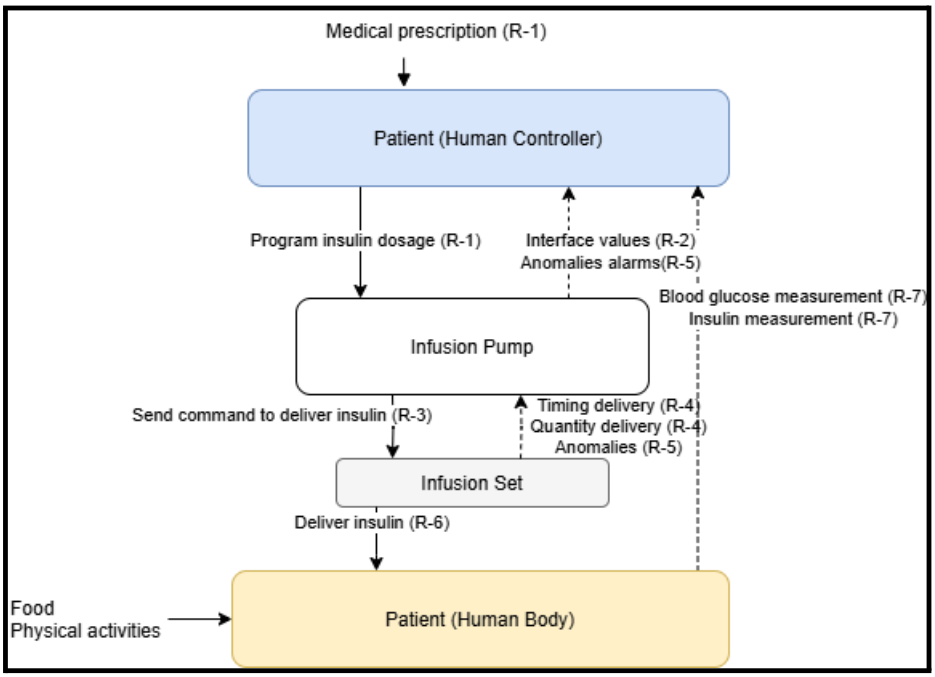
- **SD Model**



- **SR Model**


E1 - Patient (Human Controller)
Use the IIP
Have programmed insulin dosage
Meaure blood glucose
Measure insulin
<<safetyGoal>>
SC-01: The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
<<safetyGoal>>
SC-02: The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]
<<safetyGoal>>
R-02: Verify that the device interface confirms the programmed dose before administration [SC-01]
<<safetyGoal>>
R-01: Ensure that infusion settings are correctly configured and correspond to the medical prescription [SC-01, SC-02]
Have programmed insulin dosage
E2 - Insulin Pump
Deliver insulin for treatment
Receive insulin injection
<<safetyGoal>>
SC-01: The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
<<safetyGoal>>
SC-02: The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]
<<safetyGoal>>
R-03: Administer insulin only according to validated infusion parameters and prevent unauthorized dosages [SC-01]
<<safetyGoal>>
R-04: Monitor timing and quantity of delivery to ensure correct dose is given at the right time [SC-02]
<<safetyGoal>>
R-05: Detect anomalies (e.g., occlusions, over-delivery) and alert the user immediately [SC-01, SC-02]
Take insulin measurements
Take blood glucose measurements
E4 - Patient (Human Body)
Receive insulin treatment
Take insulin measurements
Take blood measurements
<<safetyGoal>>
SC-02: The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]
<<safetyGoal>>
R-08: Respond physiologically to insulin in a way that is consistent with treatment expectations (acknowledging variability) [SC-02]
Receive insulin infusion
E3 - Infusion set
Administer insulin
Administer insulin injection
<<safetyGoal>>
SC-01: The system must not administer insulin in excess of the prescribed dose or in unintended circumstances. [H1]
<<safetyGoal>>
SC-02: The system must ensure that the prescribed insulin dose is delivered at the correct time and in the correct amount. [H2]
<<safetyGoal>>
R-06: Maintain physical integrity to prevent leaks or unintended flow of insulin [SC-01]
<<safetyGoal>>
R-07: Ensure correct and timely delivery of insulin from pump to patient [SC-02]

Step 3 - Define the Control Structure



Step 4 - Identify Unsafe Control Actions (UCAs)

Control Action	From/To	Not Providing Causes Hazard	Providing Causes Hazard	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Program insulin dosage (R-1)	Patient / Infusion Pump	**UCA-01:** Patient does not provide "Program insulin dosage" when insulin is required, leading to underdose [H1]	**UCA-02:** Patient provides "Program insulin dosage" with a value higher than prescribed, leading to overdose [H2] **UCA-03:** Patient provides "Program insulin dosage" with a value lower than	**UCA-04:** Patient provides "Program insulin dosage" too late, leading to hyperglycemia [H1] **UCA-05:** Patient provides "Program insulin dosage" too early, leading to premature insulin	*Not applicable*

			prescribed, leading to underdose [H1]	administration and resulting in hypoglycemia [H2]	

Additional hazards cause identified independently of the STPA results

Hazard Cause
HC-01: The pump is misplaced or inaccessible to the patient.[H2]

Step 5 - Identify Controller Constraints

Unsafe Control Action	Controller Constraint
UCA-01: Patient does not provide “Program insulin dosage” when insulin is required, leading to underdose. [H1]	**C-01:** The patient must program the insulin dosage whenever insulin is required, according to clinical guidance. [UCA-01]
UCA-02: Patient provides “Program insulin dosage” with a value higher than prescribed, leading to overdose. [H2]	**C-02:** The patient must ensure the programmed insulin dosage does not exceed the value prescribed by the physician. [UCA-02]
UCA-03: Patient provides “Program insulin dosage” with a value lower than prescribed, leading to underdose. [H1]	**C-03:** The patient must verify that the programmed dosage meets the minimum prescribed threshold to avoid underdosing. [UCA-03]
UCA-04: Patient provides “Program insulin dosage” too late, leading to hyperglycemia. [H1]	**C-04**: The patient must program the insulin dosage in a timely manner, according to the prescribed administration window. [UCA-04]
UCA-05: Patient provides “Program insulin dosage” too early, leading to premature insulin administration and resulting in hypoglycemia. [H2]	**C-05:** The patient must not program the insulin dosage before the appropriate physiological or dietary conditions occur. [UCA-05]
HC-01: The pump is misplaced or inaccessible to the patient.	**C-06:** The insulin pump must always be correctly placed and readily accessible to the patient.

Step 6 - Analyze Loss Scenarios and Derive Safety Requirements

UCA	Loss Scenario (LS)	Safety Requirement (SR)

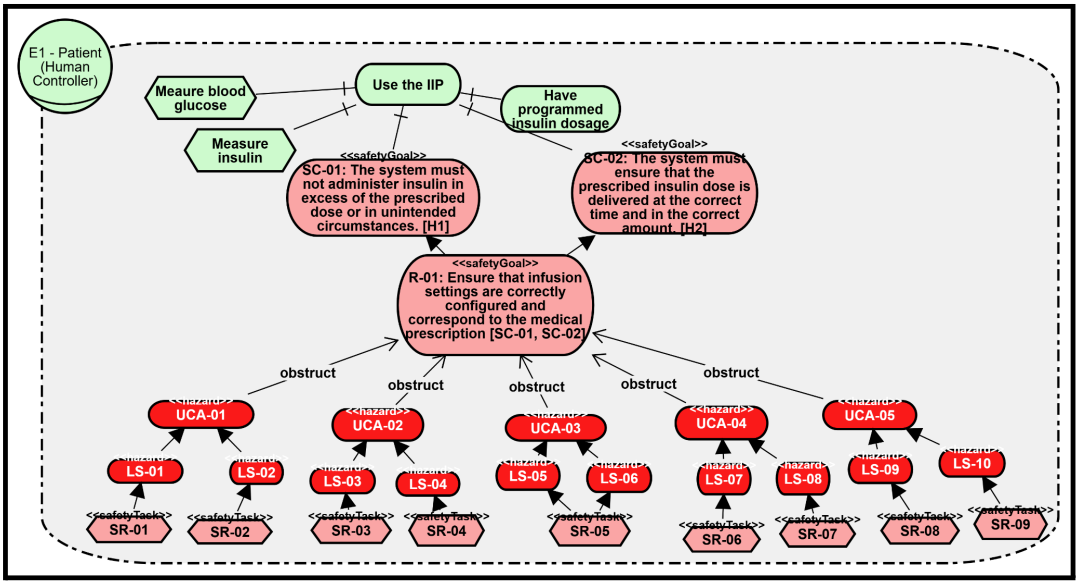
UCA-01: Patient does not provide “Program insulin dosage” when insulin is required, leading to underdose [H1]	**LS-01:** The patient forgets to program the dose after the meal, resulting in hyperglycemia. [UCA-01] *Martinazzo (2022)* **LS-02:** The system does not issue a reminder to program the dose after detecting a meal event. [UCA-01] *Ribeiro et al. (2024)*	**SR-01:** The system shall generate an alert if insulin is not programmed within 15 minutes after a meal is detected. [LS-01] *Zhang et al. (2011)* **SR-02:** The interface must maintain a visible warning if no insulin programming is detected post-meal. [LS-02] *Ribeiro et al. (2024)*
UCA-02: Patient provides “Program insulin dosage” with a value higher than prescribed, leading to overdose [H2]	**LS-03:** The patient repeats a bolus due to lack of feedback on recent insulin administration. [UCA-02] *Zhang et al. (2010)* **LS-04:** Patient misinterprets the prescribed dose and enters a value higher than medically indicated. [UCA-02] *Zhang et al. (2011)*	**SR-03:** The system shall display recent insulin activity clearly before accepting a new dose. [LS-03] *Zhang et al. (2011)* **SR-04:** The system shall cross-check manual input with prescription data and alert if excess dosage is detected. [LS-04] *Zhang et al. (2011)*
UCA-03: Patient provides “Program insulin dosage” with a value lower than prescribed, leading to underdose [H1]	**LS-05:** The patient reduces the dose to avoid hypoglycemia without clinical basis. [UCA-03] *Martinazzo (2022)* **LS-06:** The system does not notify that the entered dose is below clinical expectation. [UCA-03] *Zhang et al. (2011)*	**SR-05:** The system must recommend confirmation when the user’s dose is significantly below the recommended amount. [LS-05, LS-06] *Zhang et al. (2011)*
UCA-04: Patient provides “Program insulin dosage” too late, leading to hyperglycemia [H1]	**LS-07:** The patient delays programming due to being busy or distracted, compromising glycemic control. [UCA-04] *Martinazzo (2022)* **LS-08:** The system accepts bolus entry even after blood glucose spike already occurred. [UCA-04] *Ribeiro et al. (2024)*	**SR-06:** The interface must issue periodic prompts for pending bolus if blood glucose remains elevated and no dose is scheduled. [LS-07] *Zhang et al. (2011)* **SR-07:** The system must block bolus entries considered ineffective post-prandial, requiring physician override. [LS-08] *Ribeiro et al. (2024)*

UCA-05: Patient provides “Program insulin dosage” too early, leading to premature insulin administration and resulting in hypoglycemia [H2]	**LS-09:** Patient programs insulin and forgets to eat, leading to insulin drop without carbohydrate intake. [UCA-05] *Zhang et al. (2010)* **LS-10:** Patient assumes a meal is imminent, but it is delayed due to unforeseen events. [UCA-05] *Martins et al. (2015)*	**SR-08:** System must require user confirmation that the meal is occurring before completing bolus delivery. [LS-09] *Zhang et al. (2011)* **SR-09:** If a meal confirmation is not received within a set time, bolus must be suspended or canceled automatically. [LS-10] *Martins et al. (2015)*

Additional hazards cause identified independently of the STPA results

Hazard Cause	Loss Scenario	Safety requirement
HC-01: The pump is misplaced or inaccessible to the patient.	**LS-11**: The patient is in a critical condition and does not remember where the pump was placed.	**SR-10**: The pump must have an associated mobile application that allows a "locate pump" function to trigger an audible alarm when activated.

Step 7 - Update the iStar4Safety Models



F

Strategic Rationale Model for the Robotic Medicine Delivery System Scenario

This appendix presents the final SR model of the Robotic Medicine Delivery System, used to illustrate the application of the RESafety process. In the first part of this appendix, we include the image of the complete model, divided into two parts to allow a comprehensive overview. In the following three sections, we provide the image of each modeled actor separately, to offer a clearer and more detailed view to the reader.

F.1 Complete SR Model of the Robotic Medicine Delivery System Scenario

Figures F.1 and F.2 present the complete SR model of the robotic drug delivery system used in the hospital pharmacy context. The model is shown in two vertical segments to ensure readability.

As can be seen, the model continues to represent the three main actors, along with their dependency links as defined in the SD model, and the corresponding safety logic. This artifact is the outcome of the first iteration, **Step 7: Update the iStar4Safety Models** of the RESafety process.

In the following sections, we provide a detailed view of each actor to enhance readability and facilitate a clearer understanding of their roles and safety-related elements.

F.2 Pharmacy Employee SR Model excerpt

Figure F.3 presents the excerpt of the final SR model — that is, the result of Step 7 from the first iteration of the RESafety process applied to the Robotic Medicine Delivery System.

In this section, we present and discuss the portion of the model related to the behavior of the *Pharmacy Employee*, in order to provide a clearer understanding for the reader.

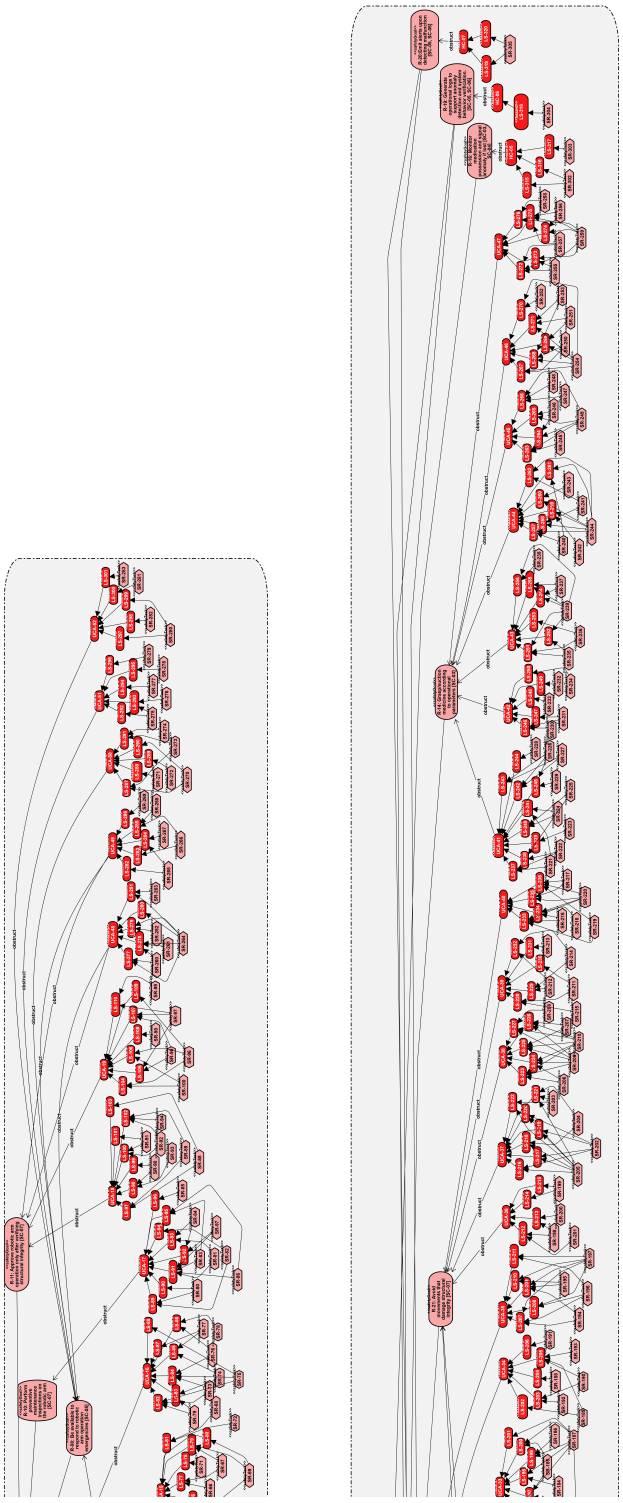


Figure F.1: SR model of the Robotic Medicine Delivery System — Part 1.

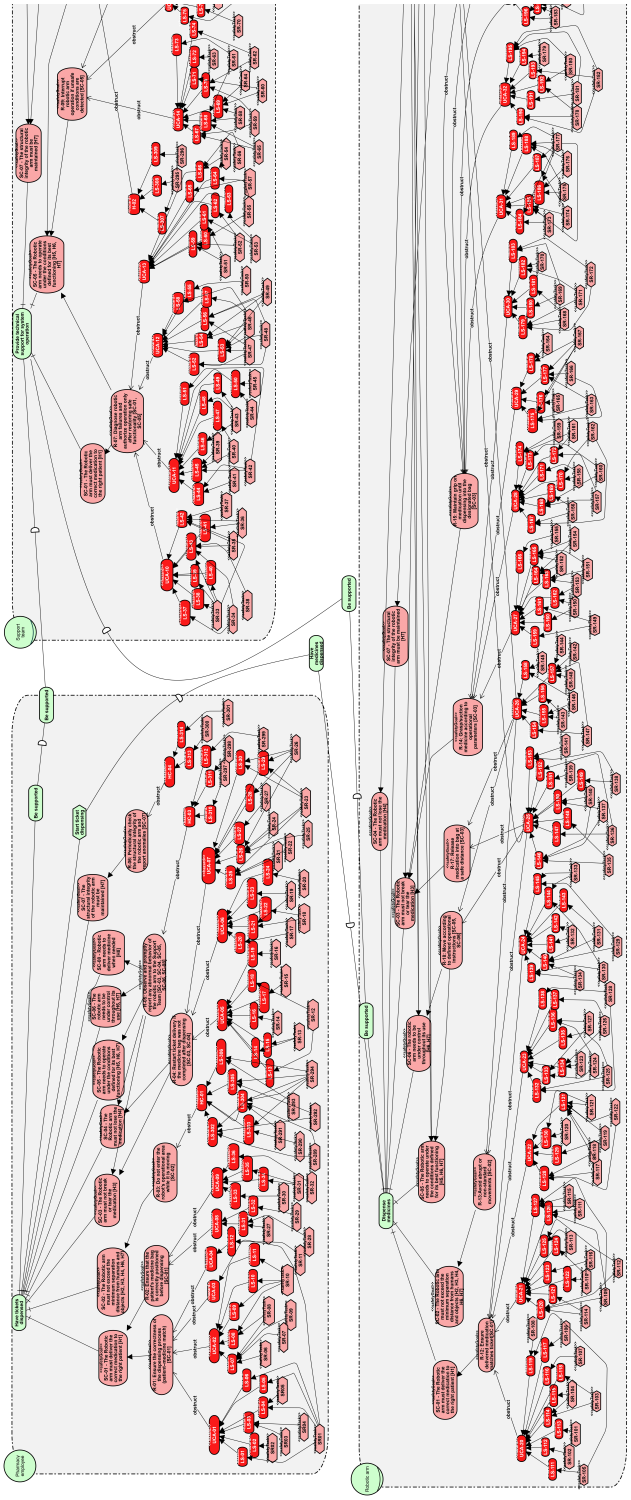


Figure F.2: SR model of the robotic drug delivery system — Part 2.

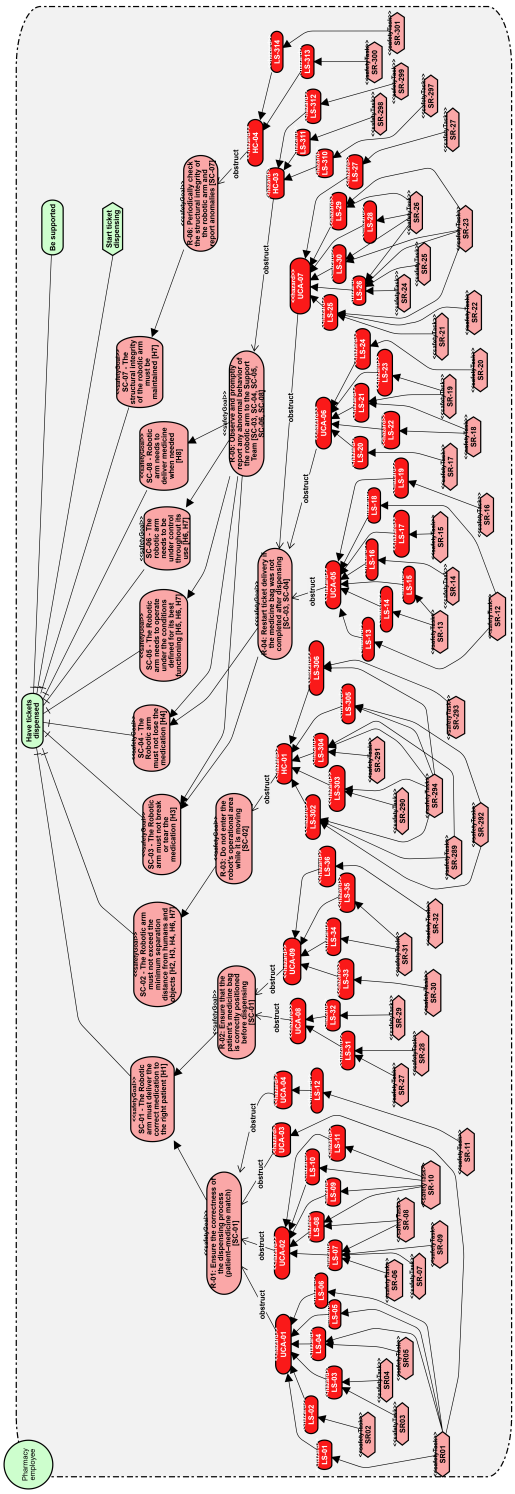


Figure F.3: SR model excerpt of the Robotic Medicine Delivery System — Pharmacy Employee.

As shown in the figure, the main goal of the *Pharmacy Employee* actor remains defined as **Have tickets dispensed**. Refining this goal, we have the *Safety Constraints* defined in Step 1.2.3, which must be satisfied by the *Pharmacy Employee* to prevent the hazards identified in Step 1.2.2 from occurring.

This actor is responsible for fulfilling, through its responsibilities defined in Step 1.2.4, all the Safety Constraints from SC-01 to SC-08. Continuing the modeling tree, each Safety Constraint (SC) was associated with the respective responsibilities of this actor, ranging from R-01 to R-06 — all of which are explicitly linked to the SCs they are intended to address.

Next, the Unsafe Control Actions (UCAs) and Hazardous Conditions (HCs) identified in Step 4 must be linked to the responsibilities they have the potential to obstruct. In other words, these are the responsibilities that, if a UCA or HC occurs, may not be fulfilled — which in turn may prevent the corresponding Safety Constraint from being satisfied. This could ultimately result in the occurrence of the associated hazard and, consequently, lead to the accident related to that hazard.

In this case, for the Robotic Medicine Delivery System, UCAs UCA-01 through UCA-09 are associated with the responsibilities assigned to the *Pharmacy Employee*. Additionally, the Hazardous Conditions HC-01, HC-03, and HC-04 are also linked to this actor's responsibilities.

At the next modeling level, all the *Loss Scenarios* that may result from the UCAs or HCs are represented. As shown, the Loss Scenarios related to the UCAs and HCs of this actor range from LS-01 to LS-36, and they are mitigated by corresponding *Safety Requirements* — represented as *Safety Tasks* — ranging from SR-01 to SR-32.

In parallel, the Loss Scenarios associated specifically with the three defined Hazardous Conditions span from LS-302 to LS-306 and from LS-310 to LS-314. The *Safety Requirements* designed to mitigate these scenarios range from SR-289 to SR-294 and SR-297 to SR-301.

The non-safety *Goal* "Be supported", on which the *Pharmacy Employee* depends with respect to the *Support Team*, is linked to the main goal of the *Support Team* actor, namely **"Provide technical support for system operation"**.

Similarly, the non-safety *Task* "Start ticket dispensing", which the *Pharmacy Employee* depends on the *Robotic Arm* to perform, is associated with the main goal of the *Robotic Arm* actor, **"Dispense medicine"**. This task is thus fulfilled through the achievement of that goal.

In the following section, we discuss the *Support Team* actor, also presenting the corresponding excerpt of the SR model that illustrates its behavior within this context.

F.3 Support Team SR Model excerpt

Figure F.4 presents the excerpt of the final SR model from the first iteration of the RESafety process applied to the modeled Robotic Medicine Delivery System.

In this section, we present and discuss the portion of the model related to the behavior of the *Support Team* in the pharmacy, in order to enhance the reader's understanding.

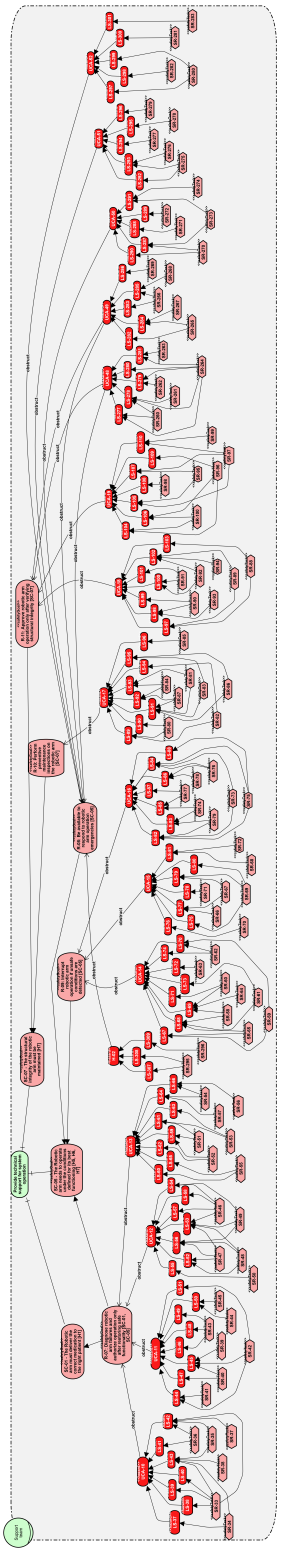


Figure F.4: SR model excerpt of the Robotic Medicine Delivery System — Support Team.

As shown in the figure, the main goal of the *Support team* actor remains defined as **Provide technical support for system operation**. Refining this goal, we have the *Safety Constraints* defined in Step 1.2.3, which must be satisfied by the *Support team* to prevent the hazards identified in Step 1.2.2 from occurring.

This actor is responsible for fulfilling, through its responsibilities defined in Step 1.2.4, the Safety Constraints SC-01, SC-05 and SC-07. Continuing the modeling tree, each Safety Constraint (SC) was associated with the respective responsibilities of this actor, ranging from R-07 to R-11 — all of which are explicitly linked to the SCs they are intended to address.

Next, the Unsafe Control Actions (UCAs) and Hazardous Conditions (HCs) identified in Step 4 must be linked to the responsibilities they have the potential to obstruct. In other words, these are the responsibilities that, if a UCA or HC occurs, may not be fulfilled — which in turn may prevent the corresponding Safety Constraint from being satisfied. This could ultimately result in the occurrence of the associated hazard and, consequently, lead to the accident related to that hazard.

In this case, for the Robotic Medicine Delivery System, UCAs UCA-10 through UCA-19 and UCA-48 through UCA-52 are associated with the responsibilities assigned to the *Support Team*. Additionally, the Hazardous Condition HC-02 is also linked to this actor's responsibilities.

At the next modeling level, all the *Loss Scenarios* that may result from the UCAs or HCs are represented. As shown, the Loss Scenarios related to this actor range from LS-37 to LS-110 and from LS-277 to LS-301, and they are mitigated by corresponding *Safety Requirements* — represented as *Safety Tasks* — ranging from SR-33 to SR-100 and from SR-260 to SR-283.

In parallel, the Loss Scenarios specifically associated with the Hazardous Conditions span from LS-307 to LS-309, and the *Safety Requirements* designed to mitigate these scenarios are SR-295 and SR-296.

For this actor, as can be observed, there was no need to model any dependencies on other actors.

Finally, in the following section, we discuss the *Robotic Arm* actor, also presenting the corresponding excerpt of the SR model that illustrates its behavior within this context.

F.4 Robotic Arm SR Model excerpt

Finally, we present the analysis of the SR model for the last actor, the *Robotic Arm*, shown in Figures F.5 and Figure F.6. The figure presents the excerpt of the final SR model from the first iteration of the RESafety process applied to this actor.

As shown in the figures, the main goal of the *Robotic Arm* actor remains defined as **Dispense medicines**. Refining this goal, we have the *Safety Constraints* defined in Step 1.2.3, which must be satisfied by the *Robotic Arm* to prevent the hazards identified in Step 1.2.2 from occurring.

This actor is responsible for fulfilling, through its responsibilities defined in Step 1.2.4,

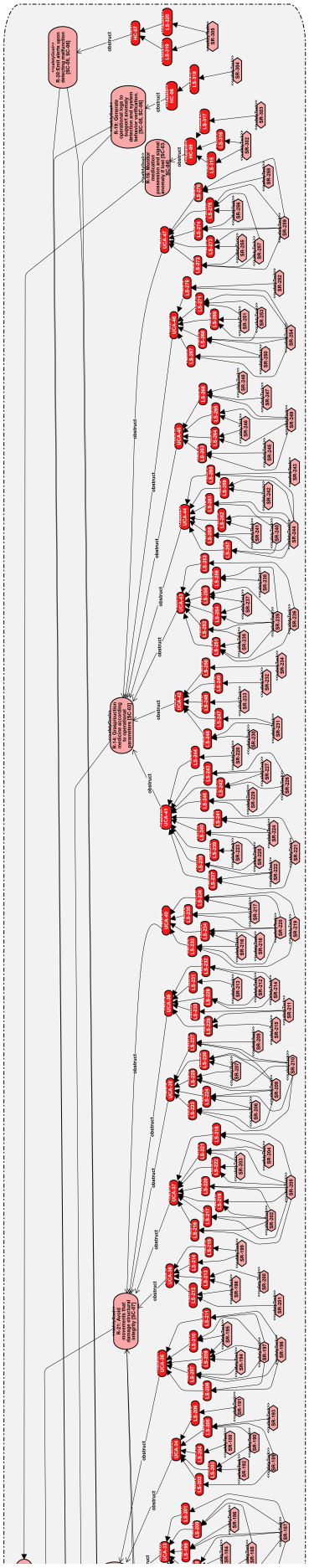


Figure F.5: SR model excerpt of the Robotic Medicine Delivery System — Robotic Arm Part 1.

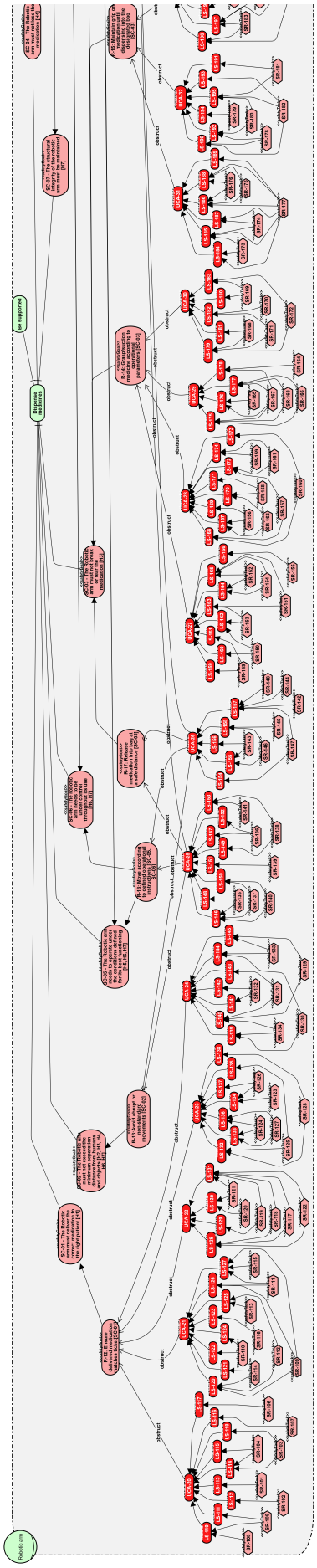


Figure F.6: SR model excerpt of the Robotic Medicine Delivery System — Robotic Arm Part 2.

the Safety Constraints SC-01 to SC-07. Continuing the modeling tree, each Safety Constraint (SC) was associated with the respective responsibilities of this actor, ranging from R-12 to R-22 — all of which are explicitly linked to the SCs they are intended to address.

Next, the Unsafe Control Actions (UCAs) and Hazardous Conditions (HCs) identified in Step 4 must be linked to the responsibilities they may obstruct. In other words, these are the responsibilities that, if a UCA or HC occurs, may fail to be carried out — which, in turn, may lead to the violation of the corresponding Safety Constraint. This violation could ultimately trigger the associated hazard and, consequently, result in the accident related to that hazard.

In this case, for the Robotic Medicine Delivery System, UCAs UCA-20 to UCA-47 are associated with the responsibilities assigned to the *Robotic Arm*. Additionally, Hazardous Conditions HC-05 to HC-07 are also linked to this actor's responsibilities.

At the next modeling level, all the *Loss Scenarios* that may result from the UCAs or HCs are represented. As shown, the Loss Scenarios related to this actor range from LS-111 to LS-276, and they are mitigated by corresponding *Safety Requirements* — represented as *Safety Tasks* — ranging from SR-101 to SR-259.

In parallel, the Loss Scenarios specifically associated with the Hazardous Conditions HC-05 to HC-07 span from LS-315 to LS-320, and the *Safety Requirements* designed to mitigate these scenarios are SR-302 to SR-305.

For this actor, it was necessary to model the non-safety goal *"Be supported"*, which is fulfilled through a dependency on the *Support Team* actor.

Thus, we conclude the explanation of the iStar4Safety Strategic Rationale model, which represents the final artifact of the first iteration of the safety analysis conducted using the RESafety process for the real Robotic Medicine Delivery System in a hospital pharmacy. This model was used to illustrate the application of our proposed approach.

It is important to note that, as a design choice, several other non-safety elements could have been modeled. However, to fulfill the illustrative purpose of our study, we consider that this iteration encompasses the necessary elements for the scope of this work.

G

Open-ended Question Responses from the Survey

This appendix presents Table G.1, which contains the complete open-ended responses from the survey conducted with experts in requirements engineering and safety engineering. It is important to note that Participant P12 is not included in this table because they did not respond to the question, which was optional.

Table G.1: Open-ended Responses by Participant, Area, and Statement (in English)

Participant	**Area**	**Open-ended Response (EN)**
P1	Requirements Engineering	"One of the strengths of the process is the way it integrates well-established modeling techniques (such as iStar) with safety analysis concepts, such as control structures. The iterative nature is also an important aspect, as it allows for progressive refinement, compatible with real-world software development cycles. As a suggestion for improvement (or future work), I believe it would be useful to offer additional examples or a catalog of examples. Overall, I consider this methodology to be a valuable contribution to the field of requirements engineering focused on the safety of critical systems."
P2	Requirements Engineering	"Answering the questionnaire solely based on the videos is a bit difficult. While watching the videos, some doubts arose that made it difficult to answer the questionnaire. For example, when answering about the ease of use of the proposed method, just by watching the videos, one cannot say the method is easy to use. However, perhaps using the method in a real, even if limited, situation, the answer could be different as the method might prove to be easy to use. One doubt that arose: what criterion should be used to start a new iteration? This was not clear from the presentation in the videos. Regarding the increased productivity using the method, it is also difficult to answer with certainty just by having a general view of the method from the videos. At first, as we are not yet used to the method, productivity will probably decrease, but after some time it may increase. This aspect will certainly be related to the mental effort required, so based only on the videos, I would say that using the proposed method requires a great mental effort, as we are not yet familiar with the steps and activities proposed in the method, although I personally know both iSTAR and STPA. With these comments, I mean that the method evaluation based only on the videos will probably be different when the user can apply it in a real or practical situation of their daily life. Therefore, I understand that after this static evaluation, based on the questionnaire, a real case study must be developed and the case study results should be compared with the results of this study based on questionnaire responses."
P3	Requirements Engineering	"One of the innovations of RESafety is its combination with goal-oriented modeling. This might be a challenge for adopting the approach, as many people are unfamiliar with or do not use iStar. Regarding efficiency or productivity, I marked neutral responses because it is difficult to foresee performance without practical use. The process is well documented and easy to follow."

Participant	Area	Open-ended Response (EN)
P4	Requirements Engineering	"From what I saw, I believe a part was missing where the obstacles come from dependencies. According to the video, the cases are analyzed for each actor, but what about those derived from dependencies?"
P5	Requirements Engineering	"Both STPA and iStar4Safety require specific technical knowledge. Inexperienced users may abandon the tool out of frustration if there is no well-structured and detailed guidance material to support them. Additionally, Step 3 (Defining the Control Structure) in STPA is essential for the remainder of the process. Correctly modeling the control structure demands knowledge in systems engineering, automation, or software architecture, which can be a significant barrier for requirements engineers. It is not clear how following the iStar4Safety model could support this step in practice. Although the proposed process includes sub-steps to assist the user, this stage is likely to be one of the most complex in the workflow."
P6	Requirements Engineering	"The presentation of the RESafety process as a BPM flow, detailing each subprocess and highlighting input and output artifacts, is a correct decision as it greatly aids in understanding the steps and the process as a whole. Especially since it is a method with many steps and tasks based on STPA, I believe this presentation format could help even analysts without prior experience in safety analysis or with the STPA method. As a Requirements Engineer with experience in STPA, I didn't have major issues understanding the process and the integration between the approaches. I believe this understanding was more natural for me due to that experience (with STPA), but I don't think it would prevent understanding by an analyst without such experience. Although I'm not an iStar specialist, I found the integration of iStar4Safety with STPA very interesting and see potential in combining these techniques to increase the expressiveness of the safety analysis by adding important and complementary information. Regarding the integration between iStar4Safety and STPA, given my inexperience with iStar, I initially had difficulty understanding exactly where and how the SD and SR models fit with the STPA analysis artifacts in Step 2. This became clearer when Step 3 of the process was presented. One suggestion, from someone who just got to know RE4Safety and the proposed integration, would be to further highlight these connections and the relationships between the artifacts in the process, perhaps using a diagram or scheme in addition to the BPM process itself to illustrate the integration. Analyzing the detail of the approach, I believe it would be interesting, as future work, to also evaluate its potential use by requirements specialists (with or without safety experience) who are unfamiliar with iStar and/or STPA, to analyze the perception of different profiles regarding the proposed process. I think the level of detail in the steps could benefit even less experienced requirements analysts in understanding and executing the safety analysis process, despite the inherent complexity of STPA. Another suggestion for future work, considering the level of detail the proposal has achieved, is that developing a tool to support the process execution could greatly contribute to its adoption. The process and proposed artifacts are already a solid base to guide the analysis, but considering the number of steps and tasks to be performed, as well as all the artifacts to be produced and the traceability between them (which is very important and needs to be considered and managed), I believe a tool could support and facilitate the tasks and further increase the intention to use the proposal, in addition to enabling its use by different analyst profiles."
P7	Requirements Engineering	"I missed a modeling based on SPEM 2.0 notation for software and systems development processes. Looking at your process, I see other professional profiles acting on RESafety. Shouldn't there be at least support from a domain expert specified in the process? For example, in scope definition! The user profile Analysts (video 1) seemed too generic: an analyst specialized in safety, right? It wasn't clear whether the iStar modeling constructors were extended to represent the safety elements from the STPA perspective (hazards, threats, risks, unsafe control actions). Are Steps 3.6 and 3.7 optional? There might be no sensors and actuators (video 2). It would add even more value if you provided a list of likely errors that safety analysts might make using your approach and HOW they could prevent them."

Participant	Area	Open-ended Response (EN)
P8	Requirements Engineering	"In the video, there is no explanation of the semantics of the link between «SafetyGoal» SC and «SafetyGoal» R (Step 2), and also between UA, LS, and SR (Step 7). Questions 1, 3, 10, and 11 are difficult to answer without actually using the technique. The questionnaire should ask about prior knowledge of i* and STPA. People unfamiliar with i* may lack the proper background to understand the explanations in the video. The understanding of the technique is very dependent on the understanding of i*. Since I am familiar with i*, what was new to me was the extension related to STPA. Even so, the video could include a step-by-step diagram construction for SR in Step 2. In Step 3, who provides the control structure (rectangles and flows)? Is it always provided? By whom? Steps 4, 5, 6 were the most valuable. It is necessary to highlight the advantage of using the extended i* model. And what if I wanted to use only the control structure and tables? And what if I wanted to use GSN?"
P9	Safety Engineering	"Combining two techniques (iStar and STPA) is always a challenge. The proposed process risks requiring additional analysis, increasing its complexity. Furthermore, the STPA steps have some misunderstandings. If you'd like to discuss this in more detail, we can schedule a video conference."
P10	Safety Engineering	"Coming from a specialization with STAMP/STPA, I saw a lot of coherence in the RESafety proposals. Thus, my responses to previous questions may be closely tied to my experience with STPA, so the scores I provided may be somewhat biased. In summary, between applying 'pure' STPA or RESafety, it seems that the advantages and disadvantages of each will depend on the application cases. For example, one of the biggest difficulties I encounter when performing STPA is obtaining information for accurately modeling systems, especially those already built. In a system with well-defined actors from the start, the SD and SR models can help a lot in this stage. But for systems with 'more abstract' actors, the difficulty would likely remain (I'm thinking about my reality in oil extraction applications, where much of the necessary information for a detailed model ends up being secret). Now, comparing it with more traditional analysis methods, the advantages are much clearer (as in the comparison of those methods with STPA). Finally, I saw a lot of quality in the study and commend the development!"
P11	Safety Engineering	"The well-defined artifacts represent a differential of the process. When completed, they assist and facilitate the specification of safety requirements. Another point that could be considered by the research is the issue of maturity levels for using the proposed process. Is it aligned with any safety maturity level? Does it consider characteristics of these levels? How can it be applied/benefit companies in this context?"