Integrating STPA with Safety Requirements Modeling

Moniky Ribeiro
Universidade Federal de Pernambuco
Recife, Brazil
smsr@cin.ufpe.br

Jaelson Castro
Universidade Federal de Pernambuco
Recife, Brazil
jbc@cin.ufpe.br

Ricardo Argenton
Universidade Federal do Vale do São Francisco
Petrolina, Brazil
ricardo.aramos@univasf.edu.br

ABSTRACT

Context: Requirements modeling is essential for Safety-Critical Systems because accidents are often due to inaccurate, incomplete or inconsistent requirements. The main reason of bad requirements is poor communication between safety engineers and requirements engineers. Objective: Our goal is to propose an approach that enable safety requirements to reflect the findings of the initial safety analysis phase. Method: We integrate two techniques 1) iStar4Safety, a goal-oriented requirements modeling language tailored for safety requirements and 2) STPA (System Theoretic Process Analysis), a well-recognized and accepted safety analysis technique. Results: Through this integration, our framework promises a more systematic and comprehensive approach to modeling early safety requirements. It supports the elicitation and analysis of safety concerns, fosters stakeholder communication, and underpins the development of inherently safer and more reliable critical systems. Conclusions: A real project, related to development of a low-cost Insulin Infusion Pump System - IIP, serves as example to illustrate the effectiveness of the proposed approach. Preliminary results indicates that the approach contributes to improving the visualization of the safety related information generated in the safety analysis such as the accidents, system level hazards, hazard causes, hazard mitigations, and safety requirements.

KEYWORDS

safety requirements, safety critical systems, iStar4Safety, STPA

1 INTRODUCTION

Safety-critical systems (SCSs) are systems that, when faced with specific hazards, may lead to accidents resulting in significant losses such as injury and death to human beings, economic damage, environmental harm and mission failure [6].

It is widely acknowledged that system failures and errors often stem from inadequately formulated and faulty requirements during the requirements phase [10]. Thus, it is imperative to integrate safety requirements from the project's inception, including the early requirements stage [11].

Requirements engineers often face challenges in addressing safety requirements, necessitating the involvement of safety engineers to conduct safety analysis [5, 7].

Given that requirements engineers typically lack specialized safety skills, they require structured guidance to address safety considerations from the outset. Conversely, safety engineers must employ robust modeling techniques to articulate the safety requirements identified through their analyses, as natural language descriptions are prone to ambiguity

It is widely acknowledged that system failures and errors often stem from inadequately formulated and faulty requirements during the requirements phase [6, 10], led by requirements engineers. Thus, integrating safety requirements from the project's inception, including the early requirements stage, is imperative [11].

Goal-Oriented Requirements modeling languages such as iStar, KAOS (Knowledge Acquisition Automated Specification), and GRL (Goal-oriented Requirements Language) can be used to organize and justify requirements, especially in the early stages [12]. iStar has gained widespread interest in the requirements community and has over a hundred extensions [4], including some aimed at safety modeling.

The goal of our research is to promote safety analysis during the early stages of development to identify and model safety requirements as early as possible. In this paper we outline an approach that seamlessly integrates iStar4Safety, a Goal-Oriented Requirements modeling language, tailored for safety modeling, with STPA (Systems Theoretic Process Analysis), a renowned safety analysis technique. This addresses challenges in representing safety analysis results, gaps in common requirements specification languages for Safety-Critical Systems (SCS), and promotes improved communication between safety analysts and requirements engineers.

The STPA is part of the STAMP (System-Theoretic Accident Model and Processes) model [7]. STPA presents a new way of analyzing safety, as it also considers that accidents can be caused by the interaction between system components, unlike previous techniques that focus solely on individual components, like FTA (Fault Tree Analysis), HAZOP (Hazard and Operability Study), and others [3, 16]. STPA is based on systems theory, where a system's main properties should be considered emergent. Controller modeling abstracts the roles of controlled processes and controllers and their interfaces through a model to achieve a holistic system view. Furthermore, the control model allows for analyzing control actions between components and their respective feedback.

iStar4Safety is a conservative and lightweight extension of the GORE (Goal-Oriented Requirements Engineering) iStar 2.0 language for modeling safety issues. It was developed to support the modeling of safety requirements languages[14]. Therefore, the language models requirements in a goal-oriented approach, including safety-related ones.

Our approach quite is novel as it relies on goal modeling language (iStar4Safety), using insights brought by the STPA technique defined by [7].

We illustrate our approach by applying it in a real project related to development of a low-cost Insulin Infusion Pump System - IIP, a partnership between academy and industry. The IIP constitutes a medical apparatus designed to administer rapid-acting insulin doses via a catheter positioned beneath the patient's epidermis, aimed at

managing Type I diabetes mellitus while upholding optimal glucose levels in the patient's bloodstream.

This paper is organised as follows. Section 2 presents the related works. Section 3 presents our iterative and incremental process for modeling safety requirements using iStar4Safety models and insights from STPA analysis. The application of this process is illustrated in Section 4 through a case study on an Insulin Infusion Pump System. Section 5 presents conclusions and a roadmap.

2 RELATED WORK

Sharifi et al. [17, 18] propose a method for FinTech certification, merging GRL's goal orientation with UCM's (Use Case Maps) process modeling. Functional goals identified in the GRL model are represented as UCMs for traceability. Additionally, they suggest STPA analysis and use its artifacts to update models and create assurance cases. While similarities exist between our work and theirs, we primarily focus on Safety-Critical Systems, streamlining the process using the iStar4Safety GORE language [14] with safety constructs. Unlike their reliance on UCM for process modeling, we use iStar4Safety and concepts from the STPA technique, simplifying our approach.

The SARSSi* approach outlined in [21] tried to combine the STPA technique with the iStar goal-oriented requirements modeling technique, generating a preliminary safety analysis. An essential difference is that the former uses iStar in its standard version to model safety elements, while our process relies on iStar4Safety. Moreover, their use of the STPA technique was only sketched.

Debbech [2] introduced GOSMO - Goal-Oriented Safety Management Ontology, a process for developing safety metrics grounded in UFO - Unified Foundational Ontology and Or-BAC - Organization-Based Control Access - Model. UFO provides generic concepts and relationships across domains, including safety and GORE. GOSMO emphasizes safety measures. Our approach differs as it models early safety concepts based on a previous metamodel [19, 20]. While we do not model all GOSMO elements, we identify essential ones, associating our work with previous efforts [20].

Our preliminary proposal [15] included the use of BPMN for the description of the consequences of a safety goal not being satisfied [15]. However, the approach has evolved significantly, culminating in the current work with several adaptations and improvements.

3 INTEGRATING ISTAR4SAFETY WITH STPA

Given the necessity for analysts to elicit and model safety requirements, we aim to present a process facilitating the early definition of safety requirements alongside modeling the other early system requirements.

In the sequence, we outline a serie of steps (see Figure 1) designed to integrate principles from STPA into iStar4Safety. The granularity level of this process's phases depends on the knowledge of the system. Each iteration will contribute to more complete artifacts, so it an iterative and incremental process.

3.1 Step 1: Define safety-critical system (SCS) scope

In this initial step, we must establish and/or to update the scope of the system we model. The output of this step will be the updated

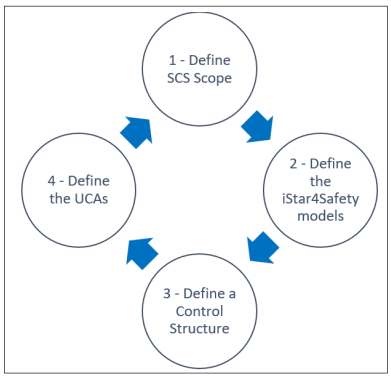

Figure 1: RESafety iterative process

safety analysis and early requirements document. This step sets the stage for subsequent phases where the analysts refine the data, drill down, and interpret it to gain insights and make decisions. During the first iteration, new artifacts and definitions will be created or used, for example, a preceding safety analysis. Furthermore, one may address some items in this step during the first iteration at a very high level or even ignoring them because it is the initial modeling iteration. In the subsequent iterations, the analysts can update, change, or insert issues that have yet to be defined early, thanks to the process's iterative and incremental nature. We define below the elements that must be considered. We divide the elements of this step into "General Concerns" and "Safety Concerns", as listed below.

(1) **General Concerns**
- Define the analysis objectives.
- System Definition.
- Gather the resources needed for analysis.
- Define the system, the system boundary, its components, and possible interactions between them.
- Define strategic stakeholders, such as persons or internal and external systems and components.

(2) **Safety Concerns**
- **Identify Accidents:** In our approach will use the **"Ax"** to label accidents. An example of an accident in an IIP is *"A1 - The patient is injured or killed from overexposure or undertreatment"*. Analysts may identify further accidents through analysis. The reference to individual components or specific causes, such as "human error", must be avoided in accidents definition [7].
- **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 accidents [7]. When identifying hazards, the analyst must specify the associated accidents, ensuring traceability between hazards and accidents. 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 accident *"A1 - The patient is injured or killed from overexposure or undertreatment"* is *"H1 - Wrong dose: Dose delivered to the patient is wrong in either amount, location, or timing [A-1]"*. In [7], the authors refine the H1 hazard into sub-hazards. We believe that sub-hazards could be the refinement of hazards, and the differentiation between them and the causes of the hazards would be the identifiers of the "hazard" type constructors.

- **Identify hazards causes:** Hazards Causes are conditions alone or associated with others, sufficient for the related hazard to occur [6, 14, 20]. Causes of hazards differ from hazards because hazards do not refer to isolated components. At this point, analysts must associate UCAs with the hazards they will lead to. You can also choose to insert other causes found by the safety analysis. An example of hazard cause for the hazard *"H1 - Wrong dose: Dose delivered to the patient is wrong in either amount, location, or timing [A-1]*" is "*UCA1 - Patient commands excessive basal dosage. [H-1]*" [8]. From the UCAs, functional requirements and safety constraints for the system can be defined. If the cause of hazard is an UCA, we can create the **"UCAx"** else, we can use the **"HCx"** identifier for the hazard cause, to meet traceability.
- **Identify hazards mitigations:** Hazard mitigation endeavors to either prevent accidents from happening altogether or to minimize their impact if they do occur. A system-level constraints define system-level conditions or behaviors that must be achieved to prevent system-level hazards. At the systems level, [7] recommends that safety constraints are not solutions related to a particular context or implementation. However, analysts must define safety constraints for the UCAs found. In our approach, safety constraints will be associated with the UCAs that they treat or with hazards found and not associated with UCAs. Examples of hazard mitigation for the UCA *"UCA01 - Patient commands excessive basal dosage [H-1]"* are *"SC1 – Patient must enter basal dose correct value according to his treatment [UCA-1]"*, and also *"SC2 - The IIP must allow an upper limit for insulin dosage [UCA-1]"*. We can use the **"SCx"** identifier for the system-level safety contraints, or even the **"Cx"** tag for constraints specifically related to components or an UCA.

Finally, consider documentation, models, safety analyses, and all pertinent artifacts associated with the system as foundational references to facilitate non-safety and safety requirements elicitation. Such reuse can alleviate the burden of the analysis process.

3.2 Step 2: Model or update the iStar4Safety models

In this step, the analyst is tasked with modeling or updating the iStar4Safety Strategic Dependency - SD and Strategic Rationale - SR models, considering the inputs from Step 1. If safety concerns were addressed in step 1, you must incorporate the new information to create/update the models here.

(1) **SD Model Create/Update:** Initially, the SD model must be modelled to include/update actors and allow better visualization of their relationships in an abstract way. Results from step 1 and other iterations should be used here. Please note that the SD model will also reflect the changes made to the SR model.
(2) **SR Model Create/Update:** Modelers need to start defining or updating the SR model based on the current information and iteration. Here, we outline the modeling process (adapted from [14]) and specify tags for each element. Elements identified by the analyst in Step 1 will have their own identifiers, along with identification for related elements, making it easier to trace and model at this stage.
   (a) Create the iStar model of the system without safety elements (non-safety related part).
   (b) Create a safety goal (ID: SGx). Here, modelers need to carefully pinpoint and depict the key objectives of the involved parties (as discussed in Step 1, *Identifying Accidents*). Then, the analyst should integrate these goals as elements within the iStar4Safety framework. Specifically, the analyst should model the objectives that were identified as accidents in Step 1 as Safety Goal elements.
   (c) Insert all system-level hazards found in Step 1 linked to the accident associated with the safety goal. Associate safety goals and hazards using the **obstruct** link. The analyst should model the hazards as "Hazard" construct with the identifier "Hx".
   (d) Identify all causes for each identified hazard. Analysts will only model the causes that are UCAS found in Step 4 after the second iteration since the UCAs will be defined at this point. In the first iteration, it is essential to define the causes of hazards through analysis, what can be done in other iterations as well, if necessary. The causes of the hazards will be the hazard-child of the hazard. For hazards with associated UCAs, consider which columns of the UCAS table have been filled. The analyst must model the Cause of the Hazard using the "hazard" construct. The hazard cause identifier will be "UCAx" if it falls into this category, or "HCx" if it is not a UCA.
   (e) Model the mitigation strategy for each leaf hazard – i.e., safety constraints found in step 1. The elements used for its modeling will be the safety tasks and safety resources. The analyst should model the safety constraint with the safety task element associated with the identifier "SCx" if it is a system-level safety constraint or as "Cx" if it is a controller constraint, a constraint generated for the UCA. Safety Tasks constructors can be used associated with Safety Resources, according to the need.
   (f) Associate the mitigation strategy with an actor which will be responsible for its achievement through dependency links.

Guidelines "a" through "f" can be repeated for successive iterations until all safety requirements are met. More information on how to use iStarSafety and modeling examples can be found at [13, 14].

3.3 Step 3: Define a Control Structure for the SCS

To define actors, limits, and functionality, researchers must create/update a control structure representing the exchanges among actors constituting the system. With the data available from the ongoing iteration, it becomes imperative to model/update elements within the control structure. The analyst must then model/update the elements described below, if feasible, considering the current iteration.

- Actors;
- Control actions carried out by the actors;
- Control Actions carried out between actors;
- Feedback between actors;
- Controller control algorithm;
- Controller process model.

The inputs for developing the control structure must stem from the outcomes of the preceding phases: the groundwork laid for system modeling (Step 1) and the iStar4Safety SD and SR models (Step 2). The arrangement of elements within the control structure must adhere to the following sequence [7]:

(1) Represent the actors of the iStar4Safety models as components of the hierarchical control structure;
(2) Decompose the components into their inherent sub-components based on the existing understanding of the problem;
(3) Establish the software module component and decompose it into sub-modules;
(4) Specify the interactions (control actions and feedback) among the components or sub-components directly linked.
(5) [OPTIONAL] At this point, the analyst may introduce new components, such as users, hardware/mechanical devices, hardware/software subsystems, and establish new relationships if necessary.

3.4 Step 4: Define the UCAs in the Control Structure

Having the control structure at hand, the requirements engineers must define through analysis which control actions may require attention in terms of safety, then being treated as UCAs [7]:

- Control actions that may be unsafe:
  - Those that, if not carried out, could lead to accidents;
  - Those that, if carried out, could lead to accidents;
  - Those that, if provided ahead of time or even after time, could lead to accidents;
  - And, finally, those that are provided for a long or short time could lead to accidents.

As stated in [7], the definition of UCAs serves to pinpoint behaviours that analysts must prevent. Each UCA must be linked with one or more hazards at the system level; analysts outline these hazards in Step 1. Consequently, when identifying UCAs, analysts must ascertain their association with specific hazards, thereby ensuring traceability between UCAs and hazards. For UCAs not yet linked to any identified hazard, analysts should contemplate updating the list of hazards, in the next iteration, to incorporate the new hazard revealed through that UCA. Our process follows an iterative methodology. Hence, analysts will revise the artifacts at each step during subsequent phases, as required.

Upon conclusion of step 4, other iterations can occur (return to step 1), allowing the analysis to be refined.

4 PROCESS ILLUSTRATION

We showcase our process by modeling an IIP system. Note that our aim is not to provide a complete model but rather to provide an illustration of our approach, mainly due to space constraints. Below, we outline the steps involved.

4.1 Step 1: Define safety-critical system (SCS) scope

We outline the process undertaken in the initial iteration, denoted as step 1. The process is presented below.

(1) **General Concerns**
- Define the analysis objectives: This analysis aims to model an IIP through iterative processes, producing updated versions of the system analysis document.
- System Definition: The IIP, a Safety-Critical System, is designed to aid in treating type 1 Diabetes Mellitus. Automated IIP offers enhanced treatment flexibility by automating multiple stages of the infusion process, mimicking human physiological responses. These pumps administer rapid-acting (bolus) and continuous (basal) insulin dosages.
- Resources needed for analysis: Articles [1, 8, 9, 22, 23], Books [6, 7] and other ones, like standards, manuals.
- Define the system, the system boundary, its components, and possible interactions between them.
  - System: Insulin Infusion Pump System – IIP.
  - System Boundary: We will consider in this analysis the IIP and stakeholders directly involved in the use of the pump.
  - Components:
    - Patient / Infusion Pump / Infusion set / External environment
    - Interactions between components: To define. In subsequent iterations, analysts will refine components and subcomponents, including Control Buttons, LCD/Audio System, Microcontroller, Stepper Motor, Stepper Motor Driver, Mechanical Transmission, and Insulin Syringe within the IIP system.
- Strategic stakeholders:
  - Patient / Medical support / Technical support / Manufacturers

(2) **Safety Concerns**
- **Identify Accidents:**
  - A1 - The patient is injured or killed from overexposure or undertreatment
  - A2 - Loss of equipment reputation

- **Identify system-level hazards:**
  - H1 - Dose delivered to the patient is wrong in quantity, location or timing [A1] [A2]
- **Identify hazards causes:** To better illustrate our process, consider that analysts identified Hazard Causes UCA1 and UCA2 after the initial iteration, specifically during the second iteration when the control structure model was available for reference.:
  - UCA1 - Do not send command for basal dosage [H1]
  - UCA2 - Commands excessive basal dosage [H1]
  - HC1 - Broken insulin valve, delivering free flow of insulin [H1]
- **Identify hazards mitigations:**
  - C1 - Command for basal dosage needs to be sent [UCA1]
  - C2 - Commands for basal dosage need to be correct [UCA2]
  - C3 - The insulin valve needs regular checks to ensure it is working properly [HC1]

4.2 Step 2: Model or update the iStar4Safety models

Figure 2 represents the SD model based on the control structure depicted in Figure 4[1]. Control actions external to the system applied to it were associated with the "External environment" actor. The SR model encompasses all actors and their interrelationships, along with the details of each actor. Due to space constraints, for the purpose of exemplification, we only depict a subset of the original model.

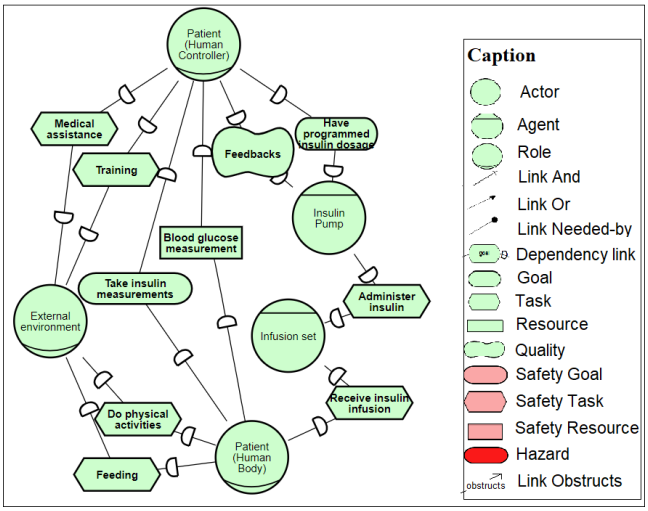

Figure 2: Pump Insulin SD Model

In Figure 3, the Insulin Pump actor was chosen to serve as a representative component. Note that **SG1** and **H1**, if we consider the first iteration, were defined in step 1. The analyst can refine them in future iterations and add causes and safety constraints.

[1]Note that artifacts generated in previous iterations can refine the SD model and the entire process. For instance, the SD model shown in Figure 2 was refined using the control structure developed in Step 3.

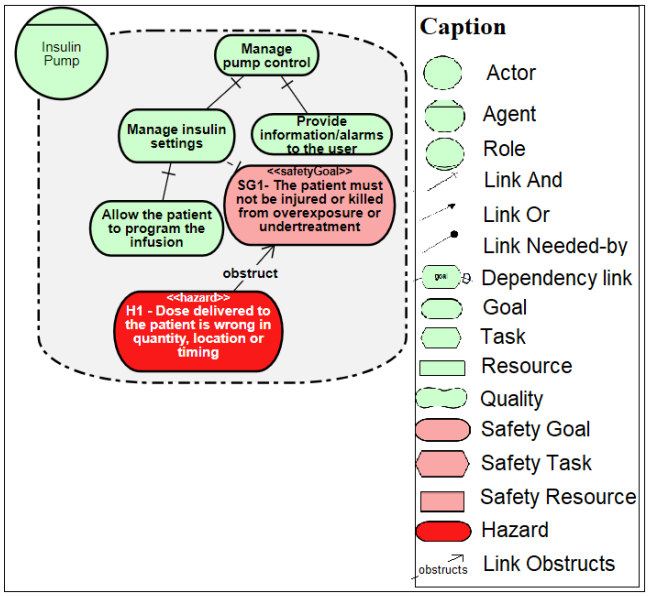

Figure 3: Excerpt of the first iteration SR model of Microcontroller Actor

4.3 Step 3: Define a Control Structure for the SCS

Figure 4 shows the model created for the IIP structure. This control structure delineates the flow between components in an IIP. At this stage, we adopt a high-level control structure approach, aligning with early IIP modeling. Following STPA guidelines, we initially establish an abstract control structure, progressively refining it over iterations [7].

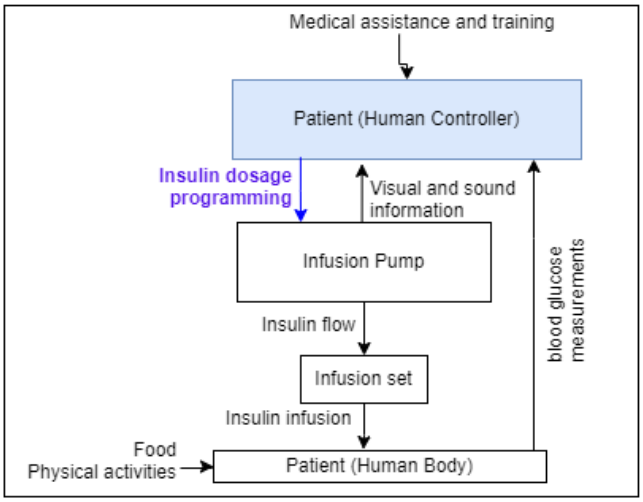

Figure 4: A high-level Insulin Infusion Pump System Controller Model.

Table 1: UCAS DEFINITION

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Insulin dosage programming	UCA-1: Do not send command for basal dosage	UCA-2: Commands excessive basal dosage	UCA-3: Commands basal dosage for each hour of the day in the wrong order	UCA-4: Commands extended bolus too short

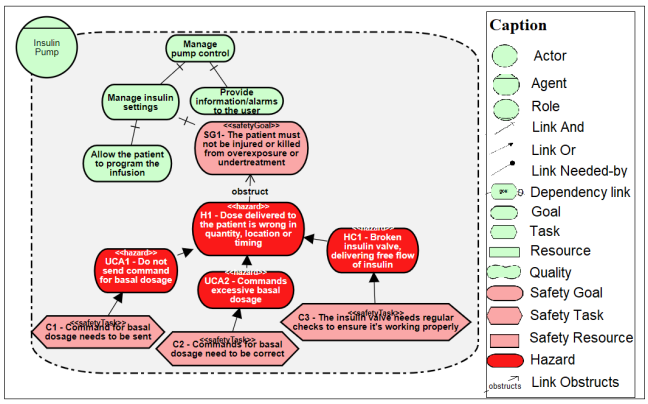

Figure 5: Excerpt of the SR model of Microcontroller Actor in second iteration.

4.4 Step 4: Define the UCAs in the Control Structure

When analysing the control structure presented in Figure 4 and comparing it with the knowledge of analysts and artifacts, we find that the control action *"Insulin dosage programming"* can be unsafe in all four situations described in Step 4.

Thus, we generated Table 1, detailing in which situations this action can be an UCA. Note that due to space constraint only a subset of the conceivale UCAs was presented.

With the results found in Step 4, return to Step 1, if it is necessary to further refine the models. Figure 5 shows the updated SR model with the Causes of hazards and mitigations (safety constraints) found during the current iteration. The analyst will implement this SR model update during Step 2 of the next iteration.

5 CONCLUSIONS AND ROADMAP

This paper presents a novel approach that seamlessly integrates iStar4Safety, a Goal-Oriented Requirements modeling language, with STPA, a safety analysis technique. It enables safety requirements to reflect the findings of the initial safety analysis phase. It provides structured guidance to address safety considerations from the outset and offers a robust modeling technique to articulate the safety requirements identified through the safety analyses. As a result it will improve communication between safety engineers and requirements engineers. Moreover, the approach is domain-independent, so it can be applied to different application contexts of safety critical systems.

Our process hopes to improve safety concern identification, stakeholder communication, and system reliability. It will enable requirements analysts to address safety issues early using a familiar language, while providing safety engineers with advantages, such as integrating their safety analysis with a requirement modeling technique.

We are currently investigating how the concept of Loss Scenarios, present in STPA analysis, could be integrated into our approach. One possibility is to model the Loss Scenarios using process modeling languages like BPMN.

Furthermore, we aim to explore how qualities like security, reliability, and privacy impact the system and interact with safety properties.

Our work has several limitations. It requires prior knowledge of STPA and iStar4Safety, though a single 2-hour instructional session on iStar4Safety was previously found sufficient [13]. Our analysis is based on a single example of a safety-critical system due to space constraints, resulting in only partial models that highlight critical system characteristics. Additionally, our involvement may introduce confirmation bias.

For future work, we aim to address these limitations by applying our approach to multiple case studies and seeking validation from safety and requirements engineering professionals. This will help generalize our findings and substantiate our claims more robustly.

We plan to apply our approach in a real case study related to the use of a robotic system in hospitals. Furthermore, we aim to explore how qualities like security, reliability, and privacy impact the system and interact with safety propertiies.

ACKNOWLEDGMENTS

The authors would like to express their gratitude for the financial support provided by CNPq, CAPES and FACEPE. Their support has been instrumental in the successful completion of this research.

REFERENCES

[1] Esra Bas. 2020. STPA methodology in a socio-technical system of monitoring and tracking diabetes mellitus. *Applied Ergonomics* 89 (2020), 103190. https://doi.org/10.1016/j.apergo.2020.103190

[2] Sana Debbech, Philippe Bon, and Simon Collart-Dutilleul. 2019. Conceptual Modelling of the Dynamic Goal-oriented Safety Management for Safety Critical Systems. In *Proceedings of the 14th International Conference on Software Technologies* (Prague, Czech Republic) *(ICSOFT 2019)*. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, 287–297. https://doi.org/10.5220/0007932502870297

[3] S. Fugivara, A.V.D. Merladet, and C.H.N Lahoz. 2021. STPA Analysis of Brazilian Sounding Rockets Launching Operations. *Microgravity Sci. Technol.* 33, 43 (jun 2021). https://doi.org/10.1007/s12217-021-09871-x.

[4] Enyo Gonçalves, Marcos Antônio de Oliveira, Ingrid Monteiro, Jaelson Castro, and João Araújo. 2018. Understanding what is important in iStar extension

proposals: the viewpoint of researchers. *Requirements Engineering* (20 Jul 2018). https://doi.org/10.1007/s00766-018-0302-5

[5] John C. Knight. 2002. Safety Critical Systems: Challenges and Directions. In *Proceedings of the 24th International Conference on Software Engineering* (Orlando, Florida) *(ICSE '02)*. ACM, New York, NY, USA, 547–550. https://doi.org/10.1145/581339.581406

[6] Nancy G. Leveson. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Mit Press, Massachusetts, London, England.

[7] Nancy G Leveson and John P. Thomas. 2018. *STPA Handbook* (first ed.).

[8] Aldo Martinazzo. 2022. Gerenciamento de risco de uma bomba de infusão de insulina de baixo custo (in English: Risk management of a low-cost insulin infusion pump).

[9] Luiz Eduardo G. Martins, Hanniere de Faria, Lucas Vecchete, Tatiana Cunha, Tiago de Oliveira, Dulce E. Casarini, and Juliana Almada Colucci. 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 (CBMS '15)*. IEEE Computer Society, Washington, DC, USA, 338–343. https://doi.org/10.1109/CBMS.2015.14

[10] Luiz Eduardo G. Martins and Tony Gorschek. 2017. Requirements Engineering for Safety-Critical Systems: Overview and Challenges. *IEEE Software* 34, 4 (2017), 49–57. https://doi.org/10.1109/MS.2017.94

[11] Luiz E. G Martins and Tony Gorschek. 2022. *Requirements Engineering for Safety-Critical Systems*. River, Denmark.

[12] John Mylopoulos, Lawrence Chung, and Eric Yu. 1999. From Object-oriented to Goal-oriented Requirements Analysis. *Commun. ACM* 42, 1 (Jan. 1999), 31–37. https://doi.org/10.1145/291469.293165

[13] Moniky Ribeiro. 2019. Desenvolvimento de uma extensão da linguagem de modelagem iStar para Sistemas Críticos de Segurança - iStar4Safety. (in English: Development of an extension of the iStar modeling language for Safety Critical Systems - iStar4Safety).

[14] Moniky Ribeiro, Jaelson Castro, and João Pimentel. 2019. iStar for Safety-Critical Systems. In *Proceedings of the 12th International i* Workshop co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4th, 2019 (CEUR Workshop Proceedings, Vol. 2490)*, João Pimentel, Juan Pablo Carvallo, and Lidia López (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-2490/paper15.pdf

[15] Moniky Ribeiro, Jaelson Castro, Ricardo Argenton Ramos, Maria Lencastre, Abimael Santos, and Oscar Pastor. 2023. Integrating Goal Oriented Requirements Modeling and Safety Analysis with Requirements4Safety. In *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 (CEUR Workshop Proceedings, Vol. 3533)*, Sotirios Liaskos, Roxana L. Q. Portugal, and Alejandro Maté (Eds.). CEUR-WS.org, 40–46. https://ceur-ws.org/Vol-3533/iStar23_paper_7.pdf

[16] Jeremiah(Jeremiah Robertson. 2019. *Systems theoretic process analysis Applied to manned-unmanned teaming*. Ph.D. Dissertation.

[17] Sepehr Sharifi, Daniel Amyot, John Mylopoulos, Patrick McLaughlin, and Ray Feodoroff. 2022. Towards Improved Certification of Complex FinTech Systems – A Requirements-based Approach. In *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*. 205–214. https://doi.org/10.1109/REW56159.2022.00046

[18] Sepehr Sharifi, Patrick McLaughlin, Daniel Amyot, and John Mylopoulos. 2020. Goal Modeling for FinTech Certification. In *Proceedings of the Thirteenth International iStar Workshop co-located with 28th IEEE International Requirements Engineering Conference (RE 2020) (CEUR Workshop Proceedings, Vol. 2641)*, Renata S. S. Guizzardi and Gunter Mussbacher (Eds.). CEUR-WS.org, 73–78. http://ceur-ws.org/Vol-2641/paper_13.pdf

[19] Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, and Tony Gorschek. 2018. Safe-RE: A Safety Requirements Metamodel Based on Industry Safety Standards. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering* (Sao Carlos, Brazil) *(SBES '18)*. ACM, New York, NY, USA, 196–201. https://doi.org/10.1145/3266237.3266242

[20] Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, Tony Gorschek, and Carla Silva. 2017. Specifying Safety Requirements with GORE Languages. In *Proceedings of the 31st Brazilian Symposium on Software Engineering* (Fortaleza, CE, Brazil) *(SBES'17)*. ACM, New York, NY, USA, 154–163. https://doi.org/10.1145/3131151.3131175

[21] Jéssyka Vilela, Carla Silva, Jaelson Castro, Luiz Eduardo G. Martins, and Tony Gorschek. 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* (Salvador). SBC, Porto Alegre, RS, Brasil, 17–24. https://doi.org/10.5753/bware.2019.7504

[22] Yi Zhang, Raoul Jetley, Paul L. Jones, and Arnab Ray. 2011. Generic Safety Requirements for Developing Safe Insulin Pump Software. *Journal of Diabetes Science and Technology* 5, 6 (2011), 1403–1419. https://doi.org/10.1177/193229681100500612 arXiv:https://doi.org/10.1177/193229681100500612 PMID: 22226258.

[23] Yi Zhang, Paul L. Jones, and Raoul Jetley. 2010. A Hazard Analysis for a Generic Insulin Infusion Pump. *Journal of Diabetes Science and Technology* 4, 2 (2010), 263–283. https://doi.org/10.1177/193229681000400207 arXiv:https://doi.org/10.1177/193229681000400207 PMID: 20307387.