iStar for Safety-Critical Systems

Moniky Ribeiro[1], Jaelson Castro[1], João Pimentel[2]

[1] Universidade Federal de Pernambuco, Campus Recife, Brazil
{smsr,jbc}@cin.ufpe.br
[2] Universidade Federal Rural de Pernambuco, Brazil
{joao.hcpimentel}@ ufrpe.br

Abstract. Context: Safety-Critical Systems is a system whose failure or malfunction may lead to damage or loss of life, destruction of property, loss of missions or environmental damage. **Objective:** This work proposes iStar4Safety, an extension of the iStar 2.0 language to enable the modeling of safety requirements. **Method:** The definition of new constructs to model specific safety-related concerns was performed by analyzing the essential concepts defined by the specialized literature. Moreover, the language metamodel is proposed along with its constraint rules. **Results:** The definition of iStar4Safety, a goal-oriented requirements language that enables the modeling of safety concerns in the early stage of system development. An Insulin Infusion Pump System was used to illustrate the use of the new language. A modeling tool was developed and is available for public use. **Conclusions:** The results of a preliminary evaluation indicate that the **iStar4Safety** language is suitable for describing the requirements of safety-critical systems. Furthermore, it was considered simple and easy to use while preserving the constructs of the iStar 2.0 language.

Keywords: Requirements Engineering. Safety-Critical Systems. iStar. Safety. Extension.

1 Introduction

A Safety-Critical System (SCS) involves both hardware and software components so that, if they fail or behave unexpectedly may ensue damage to people or property, significant financial losses, damage to the environment or even loss of life [7]. Therefore, to avoid unacceptable or unwanted behaviors of these Safety-Critical Systems, more care, and rigor in their development is required when compared to traditional information systems. Software components of SCSs are in charge of critical functions in areas such as aeronautics, automotive, healthcare, robotics, power generation, among others. During the development of these systems, steps must be taken to ensure hazards' mitigation to prevent accidents.

In this paper, we propose the iStar4Safety language [10]. This new language is appropriate for the modeling of safety requirements that comply with Preliminary Safety Analysis (PSA). Thus, the safety requirements will be defined as early as possible in the development process of Safety-Critical System. The iStar4Safety extends the iStar 2.0 language by adding four new constructs and a link. We propose a metamodel for the new language in conjunction with some constraint rules, and also an illustration of the use of extension in modeling a Safety-Critical System. To facilitate the adoption of the new iStar4Safety modeling language, we have developed the piStar-4Safety tool[1], which is an extension of the piStar tool [17].

2 Related Work

The KAOS language described in [5] is a goal-oriented modeling language with a set of well-established formal analysis techniques. An important concept in the KAOS language is the idea of an obstacle that allows the representation of situations where some fact can obstruct the satisfaction of a goal, expectation or requirement. However, there are no extensions of KAOS for Safety-Critical Systems in the literature.

Secure Tropos [9] is an extension of the Tropos methodology [3] which adds the concept of security constraint, as well as extends the concepts of dependency, goal, task, and resource from the native language to address security concerns. Threats to security goals are circumstances that can cause losses, represented by threat construct. But, it is not appropriate to address safety concerns.

RiskML [11] is a modeling framework that uses conceptual modeling to assess risks in the adoption of open source software (OSS) components. The framework is based on the ability to explore OSS measures as possible risk indicators, and to relate them to higher level organizational elements. Yet, key other concepts related to safety issues are not considered.

3 Safety Requirements Extension: iStar4Safety

The iStar4Safety extension is intended to model safety requirements as early as possible during the development of Safety-Critical Systems. In a previous work [12] several key concepts required for Preliminary Safety Analysis (PSA) in Safety-Critical Systems were defined. Because the iStar 2.0 language is unable to model these concepts, we have decided to develop this new extension.

Below we present the table 1 that describes the key safety concepts that are requirements to be modeled during a Preliminary Safety Analysis (PSA) based on a literature review and opinions of safety experts [12] and how such concepts are covered by iStar4Safety.

The development of iStar4Safety consisted in the creation of this metamodel, specification of the constraint rules and definition of the concrete syntax [10]. Then, a modeling tool was adapted to provide tool support for the creation of iStar4Safety models.

[1] The piStar-4Safety tool can be found at: http://www.cin.ufpe.br/~jhcp/pistar/4safety/

Table 1 – Definition and the relationship between Preliminary Safety Analysis concepts to be modeled and iStar4Safety

PSA Concept [12]	Definition	iStar4Safety
1 - Accident	It is an unplanned and undesired event, not necessarily unexpected, which results in at least a specific level of loss [2, 7]. The accident is the consequence of a hazard.	The concept of an accident is implicit in iStar4Safety, as an accident is a consequence of a Hazard element obstructs a Safety Goal
2 - Hazard	It is a state or conditions of a given system that added to the other conditions of the environment around it will inevitably lead to an accident [7].	A Hazard element
3- Cause of Hazard	It is represented by a condition that alone or associated with others, is/are sufficient for the related hazard to occur. [2, 12, 7].	A Hazard element refining other Hazard
4 - Environmental Condition	It is a set of components and their properties, including physical, cultural, among others, that, although not part of the system, can affect their behavior.	A Hazard element
5 - Functional Safety Requirement	These are the functional requirements used to mitigate or prevent the effects of failures identified in the safety analysis.	A Safety Task element
6 - Safety strategies	These actions aim to mitigate the consequences of a possible accident. Each mitigation has a cost to its achievement, which most often involves the consumption of some resource [12, 1].	Trees of Safety Tasks and Safety Resources
7 - Resources	In the context of Safety-Critical Systems, resources are the assets required for the correct functioning of critical requirements.	A Safety Resource element
8 - Accident Impact Level	Defines how critical the accident is to the safety of the system. This level can have five values: (1) Catastrophic (2) Hazardous/Severe-Major (3) Major (4) Minor (5) No Effect.	The accidentImpactLevel property in Safety Goal element
9 - Relationship between constructs related to hazards.	Hazards should obstruct safety goals and, also, can be caused by other hazards or be mitigated by safety tasks through AND/OR refinements.	The Obstructs link and AND/OR refinements

As shown in Figure 1, the metamodel was created preserving the original iStar 2.0 metamodel [4] (yellow elements) and adding the new iStar4Safety elements (purple). We represent the metamodel by a UML class diagram following the MOF 2.4.1

standard of the OMG. **Safety Resource** is a specialization of a **resource**. The **Safety Task** element is, in turn, a **Task** class specialization. These two elements, associated or not, can mitigate the hazards, forming some safety strategies.

The **Safety Goal** and **Hazard** classes are specializations of the **Goal** class of iStar 2.0. Elements of the **Hazard** class can obstruct **Safety Goals**, represented by a new **obstructs** link. **Hazards** are conceptualized as an specialization of **goals** because they can be considered anti-goals – i.e., that are desired *not* to happen. We prefer to represent **hazards** with a **goal** specialization due to the need to demonstrate that the **hazard** is differentiated because it is an element that is opposite to the **goal**, ie, obstructs its full realization. Hence, hazards should be avoided.

Additionally, elements of the **Safety Goal** class have the **accident impact level** property that can take one of five values according to the level of the accident that will happen if the **safety goal** is obstructed.

Metamodels may need to describe further restrictions on the representation of all language specificities, generally, additional descriptions are defined in natural language or through some formal language. Because iStar4Safety is a conservative extension, we retain the iStar 2.0 constraints defined in [4] and add new constraints related to this extension.

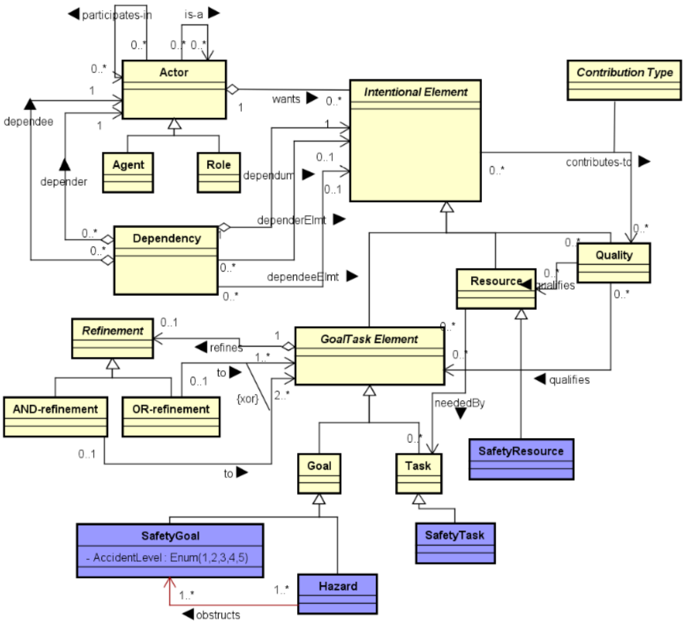

Figure 1 - iStar4Safety Metamodel.

We have defined the new constraints of the iStar4Safety metamodel in natural language:

- **Constraint rule 1** – The iStar4Safety constructs cannot be **dependum** elements.
- **Constraint rule 2 -** A **safety goal** can only be refined by **safety goals** or refined by **hazards**, and cannot be refined by both elements at the same time.

- **Constraint rule 3 -** Only **hazards-** root may be related to safety goals.
- **Constraint rule 4 -** Only **hazards**-leaf can be associated with **safety tasks**.
- **Constraint rule 5 -** Every **hazard**-leaf must have at least one associated safety strategy for him.

When defining the concrete syntax of the language we have aimed for its simplicity, that is, to use the smallest possible number of new constructs and representations in order not to hinder the learning of the language. A simple graphical representation allows modelers to create pen-and-paper diagrams without much hassle. Therefore, we chose to use the lightweight extension mechanism [15]. The textual stereotype, which is the name of the construct between the symbols "<< >>", is the lightweight option most used to represent specialized nodes [16].

For the creation of the graphical constructs, we have used the same shape of the parent classes, with different colors (light red color and dark red color), associated with stereotypes containing the name of the specialized construct. According to Figure 2, the element (A) of the figure represents the **Safety Goal** construct and the element (B) indicates the **Hazard** construct, while the element (C) represents the **Safety Task** construct and a **Safety Resource** is represented by element (D). Finally, the element (E) is the graphical representation of the **obstructs** link.

4 iStar4Safety Illustration

The Insulin Infusion Pump System was adapted from [8]. An Insulin Infusion Pump is a device intended to deliver rapid-acting insulin dosages through a catheter placed under the patient's skin to treat Type I diabetes mellitus while maintaining the adequate glucose level in the patient's blood. In this example, we have used a subset of the features.

The patient actor, represented by figure 2, has three safety goals. The "Receiving correct amount of insulin" safety goal is refined by the "Do not receive higher than correct insulin dosage" and "Do not receive lower than correct insulin dosage" safety goals.

As an example, we will describe the safety goal "Do not receive higher than correct insulin dosage". This safety goal can be obstructed by the hazard of "Insulin-free flow". The hazard of insulin-free flow can be caused by the hazard of "Valves broken in delivery path". This hazard, in turn, can be mitigated by the patient performing the safety task "Constantly check the delivery path". This task makes use of the "Delivery path" safety resource.

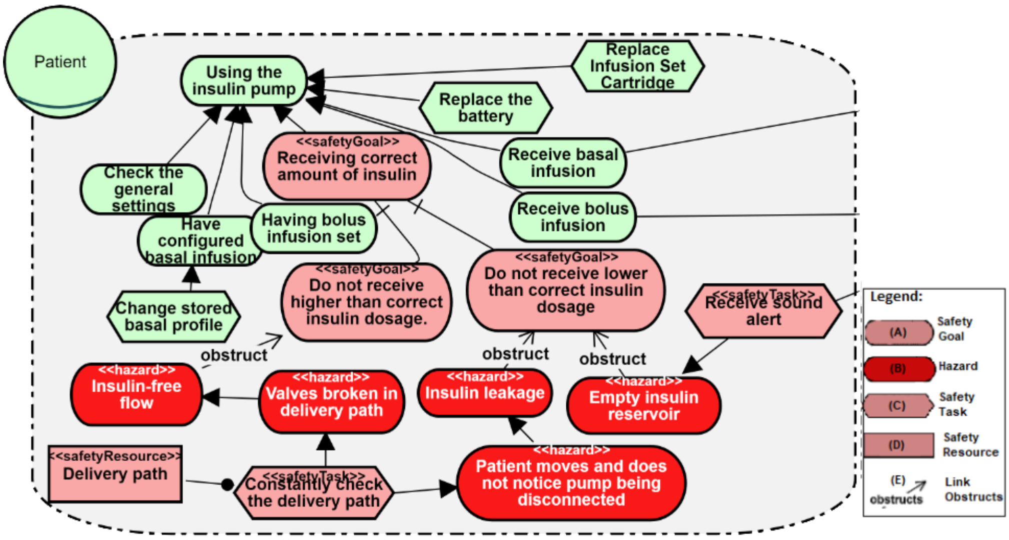
Patient
Using the insulin pump
Replace Infusion Set Cartridge
Replace the battery
<<safetyGoal>> Receiving correct amount of insulin
Receive basal infusion
Check the general settings
Receive bolus infusion
Have configured basal infusion
Having bolus infusion set
<<safetyGoal>> Do not receive higher than correct insulin dosage.
<<safetyGoal>> Do not receive lower than correct insulin dosage
<<safetyTask>> Receive sound alert
Change stored basal profile
obstruct
obstruct
obstruct
<<hazard>> Insulin-free flow
<<hazard>> Valves broken in delivery path
<<hazard>> Insulin leakage
<<hazard>> Empty insulin reservoir
<<safetyResource>> Delivery path
<<safetyTask>> Constantly check the delivery path
<<hazard>> Patient moves and does not notice pump being disconnected
Legend:
(A) Safety Goal
(B) Hazard
(C) Safety Task
(D) Safety Resource
(E) obstructs Link Obstructs

Figure 2 - Excerpt of the SR model of the Insulin Pump with safety elements.

5 Discussions and Future Work

The definition of the necessary concepts for early requirements modeling was based on our previous work [12] that identified several key concepts necessary for the Initial Analysis of Safety Requirements of Safety-Critical Systems. In order to define the elements of the new language, we have proposed the iStar4Safety metamodel, together with some constraint rules. The graphical constructs, that is, their concrete syntax, were also defined. As an additional step, a piStar-4Safety tool was proposed to support the modeling with the new language. The new language was illustrated by means of an Insulin Infusion Pump System. A quality assessment, together with expert opinion check has been made, allowing us to verify that the new language is complete, consistent, and does not conflict with the iStar 2.0 language [4] and other extensions. We have also used an empirical method to evaluate the new language [10]. Finally, the extension was approved for inclusion in the CATIE catalog of iStar extensions [6].

As future work it is suggested:

- Analyze how to model Final Requirements of Safety-Critical Systems, seeking to support to the modeling of the others Safety Analysis types;
- Implement all constraint rules in the **piStar-4Safety** tool;
- Evaluate the extension through controlled experiments [13], comparing the extension created to other forms of modeling Early Requirements of Safety-Critical Systems;

- Evaluate the Concrete Syntax developed, carrying out empirical studies;
- Reconsider how the accident impact level is represented.

Acknowledgments: The authors thank CNPq & FACEPE for their financial support.

References

1. Asnar, Y., Giorgini, P., Mylopoulos, J.: Goal-driven risk assessment in requirements engineering. Requir. Eng. 16(2), 101–116 (Jun 2011).
2. Berry, D.M.: The safety requirements engineering dilemma. In: Proceedings of the 9th International Workshop on Software Specification and Design. pp. 147–. IWSSD '98, IEEE Computer Society, Washington, DC, USA (1998)
3. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the tropos project. Information Systems 27(6), 365 – 389 (2002)
4. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. http://arxiv.org/abs/1605.07767
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20(1), 3 – 50 (1993)
6. Gonçalves, E., Heineck, T., Araújo, J., Castro, J.: CATIE: A catalogue of istar extensions. Cadernos do Ime. Série Informática, v. 48, p. 23-37, 2018.
7. Leveson, N.G.: Safeware: System Safety and Computers. ACM, New York, NY, USA (1995).
8. Martins, L. E. G.; Faria, H. d.; Vecchete, L.; Cunha, T.; Oliveira, T. d.; Casarini, D. E.; Colucci, J. A.: 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. 2015. (CBMS '15), p. 338–343.
9. Mouratidis, H., Giorgini, P.: Secure tropos: A security-oriented extension of the tropos methodology. Int. J. Soft. Eng. Knowl. Eng. 17(02), 285–309 (apr 2007).
10. Ribeiro, M.: Desenvolvimento de uma extensão da linguagem de modelagem iStar para Sistemas Críticos de Segurança - iStar4Safety (Development of an extension of the iStar modeling language for Safety Critical Systems - iStar4Safety). Master's thesis, Universidade Federal de Recife (feb 2019).
11. Siena, A., Morandini, M., Susi, A.: Modelling risks in open source software component selection. In Yu, E., Dobbie, G., Jarke, M., Purao, S., eds.: Conceptual Modeling. Volume 8824 of Lecture Notes in Computer Science. Springer International Publishing (2014) 335–348.
12. Vilela, J., Castro, J., Martins, L.E.G., Gorschek, T., Silva, C.: Specifying safety requirements with gore languages. In: Proceedings of the 31st Brazilian Symposium on Software Engineering. pp. 154–163. SBES'17, ACM, New York, NY, USA (2017).
13. Wohlin, C.; Runeson, P.; Hst, M.; Ohlsson, M. C.; Regnell, B.; Wessln, A.: Experimentation in Software Engineering. Springer, 2012.
14. Yu, E.S.K.: Modelling Strategic Relationships for Process Reengineering. Ph.D. thesis, Toronto, Ont., Canada, Canada (1995).
15. Gonçalves, E., de Oliveira, M.A., Monteiro, I., Castro, J., Araújo, J.: Understanding what is important in istar extension proposals: the viewpoint of researchers. Requirements Engineering 24, 55-84 (Mar 2019).
16. Gonçalves, E., Castro, J., Araújo, J., Heineck, T.: A systematic literature review of istar extensions. Journal of Systems and Software 137, 1 – 33 (2018).
17. Pimentel, J., Castro, J.: piStar Tool – A Pluggable Online Tool for Goal Modeling, 2018 IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, 2018, pp. 498-499.