# Specifying Safety Requirements with GORE languages

Jéssyka Vilela[12], Jaelson Castro[2], Luiz Eduardo G. Martins[3], Tony Gorschek[4], Carla Silva[2]
[1]Universidade Federal do Ceará (UFC), Brazil
[2]Universidade Federal de Pernambuco (UFPE), Brazil
[3]Universidade Federal de São Paulo (UNIFESP), Brazil
[4]Blekinge Institute of Technology (BTH), Sweden
jffv@cin.ufpe.br, jbc@cin.ufpe.br, legmartins@unifesp.br, tony.gorschek@bth.se, ctlls@cin.ufpe.br

## ABSTRACT

**Context:** A suitable representation of Safety-Critical Systems (SCS) requirements is crucial to avoid misunderstandings in safety requirements and issues in safety specification. However, current general requirements specification languages do not fully support the particularities of specifying SCS. **Objective:** In this paper, our goal is to identify and propose a set of important features that should be provided by requirements languages to support an early safety requirements specification. Moreover, we aim to compare the ability of the four most used Goal-Oriented Requirements Engineering (GORE) languages (i*, KAOS, GRL, NFR-Framework) in supporting the proposed features. **Method:** We first established a conceptual foundation and a conceptual model based on the literature, challenges elicited in previous works, and demands of safety standards at the requirements level that practitioners must satisfy in order to certify their systems. **Results**: We proposed a set of 15 features that requirements languages should provide to an early safety requirements specification. Regarding the comparison of GORE languages, in summary, all surveyed languages lacks explicit modeling constructs to express how hazards can occur in the system, the accidents, their impact and how they can mitigated. **Conclusions**: The conceptual foundation, conceptual model, and the set of features is a novelty. Finally, the features can be used to propose new requirements languages for SCS or to define extensions for the ones already available.

## CCS CONCEPTS

Software and its engineering → Software notations and tools → System description languages → System modeling languages.


SBES'17, September 20–22, 2017, Fortaleza, CE, Brazil

ACM ISBN 978-1-4503-5326-7/17/09…$15.00
https://doi.org/10.1145/3131151.3131175

**ACM Reference Format**

Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, Tony Gorschek, Carla Silva. 2017. Specifying Safety Requirements with GORE languages. In: Proceedings of SBES'17, September 20–22, 2017, Fortaleza, CE, Brazil, 10 pages.

https://doi.org/10.1145/3131151.3131175

## KEYWORDS

Goal-oriented requirements languages, Safety-critical systems, Safety analysis, Requirements engineering, Safety engineering, Goal-oriented requirements engineering.

## 1. INTRODUCTION

Safety-critical systems (SCS) are those composed of a set of hardware, software, processes, data and people whose failure can result in accidents that cause environmental damage, financial loss, injury to people and even loss of lives [27][29]. Accordingly, the development of SCS must be carefully planned and specified aiming to avoid accidents [27][29][30][31].

The increased complexity of SCS has revealed issues in safety requirements specifications such as: (i) Misunderstandings in safety requirements and specification problems; (ii) They tend to become large, ambiguous, inconsistent, and often lack clear structure affecting the process of exchanging information [32][33]; (iii) Determining the level of detail of safety specifications adequate to communication to reduce the definition of infeasible or expensive requirements to implement [34][35]. Besides, problems in the specification of SCS have been identified as a major cause of many accidents and safety-related catastrophes [5][19] [27][29].

An elaborated requirements engineering (RE) approach is crucial in the development of SCS in order to meet time, cost, and quality goals in SCS development [27]. Accordingly, safety concerns should be considered early in the development process, especially in the RE phase [1][8].

However, in safety requirements specification, there are many relationships among safety concepts, such as hazards, their causes, safety requirements and environmental conditions that must be identified and specified. Therefore, achieving an adequate representation of safety-critical systems requirements is quite fundamental for a successful safety analysis.

In order to improve the safety requirements specification, it is necessary to define a conceptual foundation as well as the features that requirements languages should have to support this task. An early safety requirements specification will contribute to: reduce the errors in requirements specifications (increasing quality) [37][38][39]; Better information presentation and increased information consistency [38][39]; and It allows exhaustive and detailed user feedback making possible to discover and specify the complete system behavior [37].

It is of paramount importance that safety concerns be handled and specified early in the development process. In this context, GORE, whose goal is to perform a domain analysis during requirements elicitation and preliminary specification, emerged as new paradigm to improve the RE process. This paradigm is based on the identification of system goals and the transformation of those goals into requirements providing a completeness criterion for the requirements specification [36], i.e. the specification is complete if all stated goals are met by the specification.

A variety of GORE languages have become popular to represent and analyze requirements. In recent systematic mapping, Horkoff et al. [21] observed that i*, GRL (Goal-oriented Requirement Language), NFR (Non-Functional Requirements) framework, GBRAM (Goal-Based Requirements Analysis Method), and KAOS (Keep All Objects Satisfied) are the most adopted languages in the selected papers. Considering the inconsistency among the terminologies adopted by safety standards, and the need of conducting preliminary safety analysis in the RE process, in this paper, we provide a set of important features that requirements languages should support for early safety requirements specification. This set is a novel contribution that can be used as criteria to guide the selection of requirements languages for specifying SCS or to propose extensions to the languages available.

Considering the proposed features, we evaluated the support of the four most used [21] GORE languages (i*, KAOS, GRL, NFR-Framework) for the safety-related concepts. By comparing these GORE languages, we observed the characteristics that make them more or less suitable for this task.

This paper is organized as follows. We provide a brief overview of the analyzed GORE languages in Section 2. We present the research methodology in Section 3. The Conceptual foundation and conceptual model for safety requirements specification in RE process is described in Section 4. In Section 5, we describe the features that should be addressed by requirements specification languages for describing safety requirements. Taking into account these features, we compare the four GORE languages in Section 6. In Section 7, we discuss some thread to validity. Related works are discussed in Section 8. Our conclusions as well as further research are presented in Section 9.

## 2. BACKGROUND

There is a variety of goal modeling frameworks, techniques, or methodologies for example i*, KAOS, GRL, NFR, GBRAM, Tropos, and AGORA. GORE languages are based on the concepts of goals, requirements, goal decomposition (division of goals into subgoals), agents (entities or processes that seek to achieve goals), tasks (represent operationalizations of goals or softgoals) [26].

The most commonly used notation for representing goal models is that of a goal decomposition tree (or graph) much in the spirit of AND/OR trees [15]. For comprehensive reviews of the major efforts undertaken along this line of research, the concepts, terminology, significance and techniques of GORE please see [13][21][36].

KAOS [24] supports different levels of expression and reasoning: semi-formal for modeling and structuring goals, qualitative for selection among the alternatives, and formal, when needed, for more accurate reasoning [13]. This formal framework involves AND and OR decompositions between goals describing desired states over entities, achieved by actions assigned to agents (Figure 1).

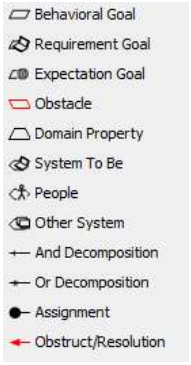


**Figure 1. KAOS syntax.**

Thus, the KAOS language combines semantic nets for conceptual modeling of goals, assumptions, agents, objects, and operations in the system, and linear-time temporal logic for the specification of goals and objects, as well as state-base specifications for operations [13].

The i* (distributed intentionality) framework [46] provides a description of work organization in terms of dependency relationships among actors [49]. The syntax of i* is presented in Figure 2. In this language, the requirements are represented by the concepts of softgoals, AND/OR decompositions, contribution links, (hard) goals, resources, and dependencies between actors (agents) [46].

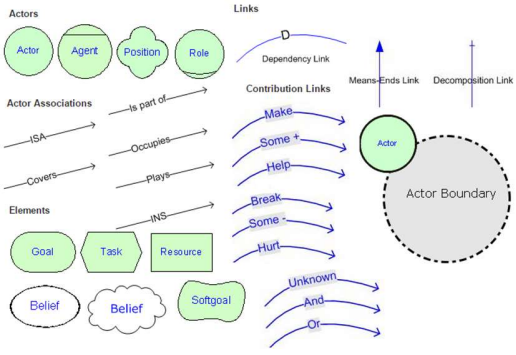


**Figure 2. istar syntax adapted from [49].**

Two models are proposed by i* framework: Strategic Dependency (SD) and Strategic Rationale (SR). The SD model captures the intentionality of the processes in the organization [13] supporting the representation of dependency relationships (goal, task, softgoal or resource) among actors. The SR model, on the other hand, allows exploring the rationale behind the dependencies in the system.

The Goal-Oriented Requirement Language (GRL) relies on a reduced set of i* [26]. GRL supports modelling and reasoning about requirements, especially non-functional requirements and quality attributes [26]. The syntax of GRL, presented in Figure 3, has three main categories of concepts: actors, intentional elements, and links [26].

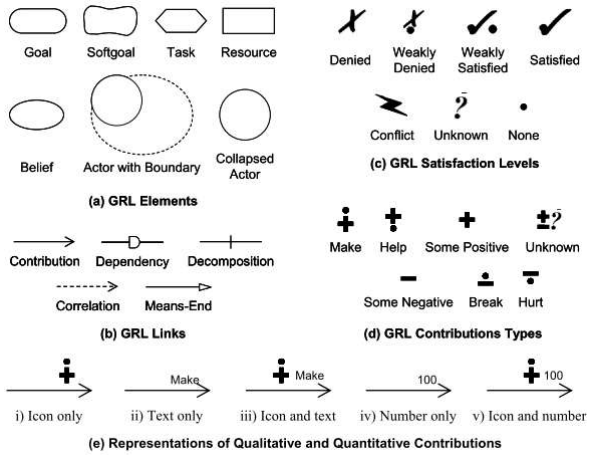


**Figure 3. GRL Syntax [26].**

According to [26], the main benefits of this language include the integration with scenarios, the support for qualitative and quantitative attributes, and a clear separation of model elements from their graphical representation, enabling a scalable and consistent representation of multiple views/diagrams of the same goal model.

The NFR framework [25] uses the concepts of softgoals (Figure 4) to represent non-functional requirements allowing to refine them through AND/OR decompositions, as well as contribution links, to represent the influences (negative and positive) to and from such goals [15].

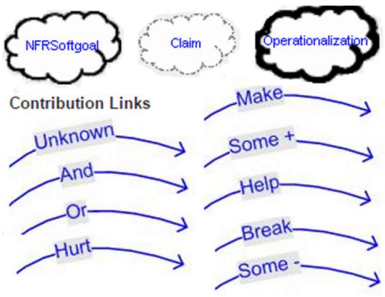


**Figure 4. NFR Syntax adapted from [25].**

The main modeling tool that the framework provides is the softgoal interdependency graph (SIG) [25]. The graphs can graphically represent softgoals, softgoal refinements (AND/OR), softgoal contributions (positive/negative), softgoal operationalizations and claims.

## 3. RESEARCH METHODOLOGY

The methodology we used to conduct this work consisted in the following steps:

1. Definition of research questions;
2. Establishment of a safety conceptual foundation;
3. Development of a conceptual model for safety requirements specification;
4. Features selection;
5. Comparison of GORE languages.

This research was guided by the following research questions:
*RQ1: What is the conceptual foundation for safety requirements specification in RE process?*

*RQ2: What are the main features that requirements languages should support in terms of safety requirements specification?*

*RQ3: What are the similarities and differences among GORE languages support for the features of RQ2?*

In order to define a conceptual foundation regarding safety-related concepts, first we conducted ad-hoc safety literature reviews as well as a systematic literature review (SLR) about RE and safety analysis integration [8]. As a result of this SLR, we proposed two taxonomies to represent the safety-related information and a detailed set of information regarding the specification of hazards.

We continued our research by conducting the analysis of safety standards from different domains as another source of information. We decided to analyze the safety standards since most of SCS should be certified and, therefore, they should be developed following their recommendations. Moreover, in interviews conducted by Martins and Gorsckek [20], the practitioners highlighted the need and importance of following an adequate safety standard.

After the comprehensive investigation of the domain, and from the elicitation of the safety conceptual constructs and the taxonomies we proposed previously, we developed a conceptual model for early safety requirements specification presented in Section 4.

The set of features that RE languages should support for early safety requirements specification was defined based on the analysis of the sources of information presented in Table 1. We adopted different kinds of source (papers, standards, books, journals).

We chose the languages to be ranked in this paper considering the mapping of Horkoff et al. [21]. We selected the top five (i*, KAOS, Tropos, NFR, and GRL). The languages are used in high number of case studies and, hence, they have more citation counts.

However, Tropos adopts the i* organizational modeling framework during early requirements analysis [22]. Hence, considering that Tropos and i* would have the same scores, in the scope of this paper, we opted to not evaluate this language. Therefore, our comparative analysis relies on four GORE languages.

**Table 1: List of sources of information.**

| # | Source | Type |
|---|---|---|
| 1 | 61508 | Generic standard |
| 2 | ISO 26262-6 | Automotive standard |
| 3 | ISO/IEC 25010 | Generic standard |
| 4 | ISO/IEC 9126 | |
| 5 | ISO 15998-1 | Machinery standard |
| | ISO 15998-2 | |
| 6 | ISO 20474-1 | Machinery standard |
| 7 | ECSS-E-HB-40A | Space standard |
| | ECSS-E-ST-40C | |
| 8 | ISO-13849-1 | Machinery standard |
| | ISO-13849-2 | |
| 9 | MIL-STD-882C | Defense standard |
| | MIL-STD-882D | |
| | MIL-STD-882E | |
| 10 | ISO/TR-14639-1 | eHealth standard |
| | ISO/TR-14639-2 | |
| 11 | Vilela et al. | SLR |
| 12 | Martins and Gorschek | SLR |
| 13 | Zoughbi et al. | Journal Paper |
| 14 | Markovski et al. | Conference Paper |

To conclude whether a language supports a feature, we relied on existing case studies as well as the description of the languages in the investigated papers and tools (see Table 2) and we did not considered extensions for these languages. To reduce possible subjectivity in the evaluation results, we had at least two people evaluating them with disagreements checked by reading again the papers adopted for each language. Disagreements were discussed among authors.

**Table 2: Papers adopted to evaluate the languages.**

| Language | Paper adopted | Tool |
|---|---|---|
| i* | [23] | OpenOme |
| KAOS | [24] | RE-Tool |
| NFR | [25] | OME |
| GRL | [26] | OME |

# 4. CONCEPTUAL FOUNDATION FOR SPECIFICATION OF SAFETY REQUIREMENTS IN RE PROCESS

Safety-critical systems can be defined as software or system operations that, if not fulfilled, fulfilled out of sequence, or incorrectly could have as consequence inappropriate control functions, or absence of control functions required for adequate system operation, that could directly or indirectly induce or enable a hazardous condition [1][2].

Terms in system safety are not used consistently [2][8]. According to Leveson [2], agreeing on terminology is always a difficult process, but it is important for communication and progress in finding solutions to problems.

In order to make clear the adopted definitions of some safety-related concepts used in this paper, and to ensure consistency, we describe them using as example the insulin infusion pump system (IIPS) [3][4]. The system goal is to provide safe and effective treatment for people suffering from Diabetes Mellitus (DM1) and to enhance the long-term health of the patients.

Figure 5 presents a conceptual model for safety requirements specification. The figure presents an abstract meta-model, which defines and relates the conceptual constructs presented in conceptual foundation. These concepts were elicited considering the most common concepts, reported in the information sources investigated, that should be specified early in the development process.

In the following, we define some of these concepts adopted in this work.

**Accident:** an undesired and unplanned (but not necessarily unexpected) event that results in (at least) a specified level of loss [2] (including loss of human life or injury, property damage, environmental pollution, and so on) [5][6]. In the IIPS, an accident can be incorrect treatment received by the patient (overdose, underdose); electrical shock; patient infection; and damage to the environment.

**Accident impact level:** the accident can have five levels of impact [10]: Catastrophic, Hazardous/Severe-Major, Major, Minor or No Effect. The incorrect treatment received by the patient has Hazardous/Severe-Major impact level.

**Environmental condition:** the state of the environment [2]. The set of factors including physical, cultural, demographic, economic, political, regulatory, or technological elements surrounding the system that could affect its safety [5]. For example, in the IIPS, an environmental condition can be Obstruction in the delivery path; the air pressure inside the pump is much smaller/larger than the air pressure environment; the pump is positioned much higher than the infusion place, causing no intentional flow.

**Hazard:** system state [6] or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss) [2] [5]. One hazard of the IIPS can be an insulin overdose; power failure due to spent battery; bad contact of sensors and actuators; parts of the machine break inside the patient's body.

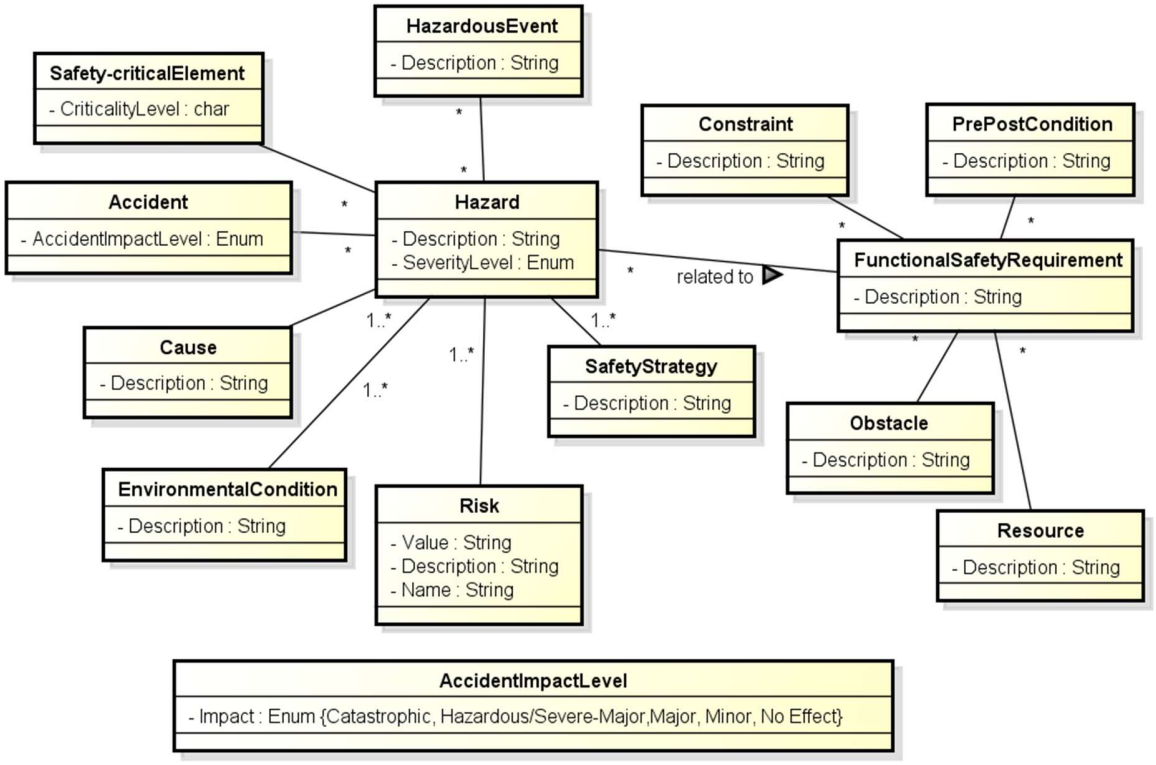


**Figure 5. Conceptual model for safety requirements specification.**

**Cause of hazard:** reason that produces hazard as effect [7]. They occur due to environmental hazard, procedural hazard, interface hazard, human factor or system cause [6][7]. A free flow is one cause of the overdose hazard as well as the presence of air in the catheter and insulin leakage.

**Safety Requirement:** is typically of the form of a quality criterion (a system-specific statement about the existence of a sub-factor of safety) combined with a minimum or maximum required threshold along some quality measure. It directly specifies how safe the system must be [6]. In the IIPS, the difference between the programmed infusion and the delivered infusion shall not be greater than 0.5%.

**Functional Safety Requirement:** The requirement to prevent or mitigate the effects of failures identified in safety analysis [1]. To mitigate the overdose hazard caused by free flow, the system can monitor the insulin reservoir.

**Safety-critical element:** can be defined as any equipment, structure or system whose failure could cause a hazard that can contribute to a major accident, or it is used in the system with the intention of avoid or reduce the accident impact level [11].

**Hazardous event:** is an event can lead to the occurrence of a hazard [11] such as the pump controller cannot monitor the status of the components.

**Risk**: is associated with Hazard and it is a combination of consequence (severity hazard) and likelihood of the hazard (risk = probability hazard x severity hazard) [9]. In the case of the insulin pump, an intolerable risk is an overdose of insulin.

**Criticality level of safety-critical element**: indicates the degree of criticality of a safety-critical element on some pre-defined scale [10]. Examples: in RTCA DO-178B [12] safety standard the categories are “A”, “B”, “C”, “D”, “E”, and in IEC 61508 [11]: “SIL 1”, “SIL 2”, “SIL 3”, “SIL 4.

**Constraint**: describes how the software must be designed and implemented providing additional information regarding requirements that must be met in order to a given goal to be achieved [13][14].

**Obstacle**: denotes the reason why a goal failed [14] consisting in behaviors or other goals that prevent or block the achievement of a given goal [13].

**Pre and post condition**: describes actions that must be executed before or after some scenario [14].

**Safety strategy**: are tactics or decisions defined to minimize the consequence or probability of the hazard [8].

**Resource**: assets, such as money, materials, staff, documents, etc., provided or used by a person or organization in order to achieve some goal [12][13].

In the next section, we present and describe the features of SCS to be supported by requirements languages.

## 5. FEATURES OF SAFETY-CRITICAL SYSTEMS

Important challenges must be addressed by requirements languages to enable an adequate representation of SCS [1][8][5][27] such as the uniformization of the information to be elicited and specified.

In this paper, we outline a set of safety concepts and features that should be supported by RE languages allowing requirements engineers to represent the results of a preliminary safety analysis (PSA). In a complete safety analysis, a richer set of attributes and relationships are specified. In this paper, we are concerned with the core concepts that are available in the RE process. Accordingly, the full list addressed of concepts used in safety analysis, such as design options and components, is out of the scope of this paper.

The high level specification of such safety concepts will be used by safety engineers as an input of a rigorous and detailed safety analysis in the preparation of reports for system certification.

We based our identification of features, that should be supported by requirements languages, in the results of the SLR we conducted [8] and in the requirements extracted from safety standards common to general industry sectors (such as ISO/IEC 61508, ISO/IEC 25010, ISO 9126, ISO 15998) whereas others are domain specific such as ISO 26262 for the automotive domain.

We depict the set of features to be supported by requirements languages for safety requirements specification in Table 3.

Requirements languages should have the ability of modeling the safety-related concepts (accidents – f1, hazards – f2, their causes – f3, environmental conditions – f4, functional safety requirements –f5). These concepts, described in Section 4, are the core information addressed in a PSA that should be specified early in the development process by requirements engineers [1][8][5].

Another important set of concepts for the specification of safety requirements are constraints (f6), obstacles (f7) and pre/post conditions (f8). The specification requirements languages must address these concepts since they may affect the functioning of some system element making it safety-critical. Accordingly, constraints, obstacles and pre/post conditions may lead to the occurrence of a hazardous event if they are not properly satisfied by the system.

The representation of safety-related concepts in a graphical way (f9) contributes to press the stakeholders to clarify system's aspects early in the design process [8][16]. Therefore, the models provide a shared meaning that engineers can use to collaborate, as when stakeholders consult a requirements specification to determine how to design a portion of the system or to perform the safety analysis [8]. Accordingly, the complex relationships of safety information are somehow mitigated with graphical representations being more consistent and less ambiguous than informal specification documents [19].

Accidents are a combination of a hazardous situation and a set of environmental conditions (context). In the analysis of the DO-178B safety standard [12], Zoughbi et al. [10] produced 54 information requirements that UML-based solution should support. Among them, it is the need of specify how a particular event affects system safety (f10). Based on the requirements of DO-178B standard and the conclusions of Zoughbi et al., we argue that a RE language should support this feature.

**Table 3: List of features for safety requirements specification to be supported by RE languages.**

| # | Feature | Source/Inspiration |
|---|---|---|
| 1 | Modeling of accident | [8][13][47][48] |
| 2 | Modeling of hazard | [8][13] [47][48] |
| 3 | Modeling of cause of hazard | [8][13] [47][48] |
| 4 | Modeling of environmental condition | [8][13] [47][48] |
| 5 | Modeling of functional safety requirement | [8][13] [47][48] |
| 6 | Representation of constraint | [13][14][15] [47][48] |
| 7 | Representation of obstacle | [13][14][15] [47][48] |
| 8 | Representation of pre and post condition | [13][14][15] [47][48] |
| 9 | Allow to represent the relationships among hazards, their causes, the environmental conditions and the functional safety requirements in a graphical form | [8][16] |
| 10 | Ability to specify how a particular event affects system safety | [10][12] [47][48] |
| 11 | Ability to specify the criticality level of safety-critical elements or the element's contributions to failure conditions | [17][18] [47][48] |
| 12 | Model and reasoning of safety strategies | [8][10][12] |
| 13 | Ability to model resources | [10][12] |
| 14 | Modeling of accident impact level | [8][10][12] [47][48] |
| 15 | Support of textual description of safety requirements | [8] [47][48] |

Safety-critical elements do not have equal magnitude or criticality to personnel safety and mission success. The consequences of hazardous events may be minimal or catastrophic depending on the system [17]. The criticality is established by analyzing the element in relation to the system and defining the level of control it exercises over functionality and contribution to accidents and hazards [18]. The determination of an element critically is part of the software system safety analysis process to define the amount of analysis and testing required to assess the software contributions to the system-level risk [18]. Therefore, a RE language should have the ability to specify the criticality level of safety-critical elements (f11).

Hazardous situations can be reduced with distinct kinds of safety mitigation strategies [8], for example, prevention, detection reaction, and adaptation, with different costs and benefits. Accordingly, modeling and reasoning of safety strategies (f12) is essential for the specification of a SCS and RE language should support it [10].

The development of safety-critical systems requires the use of many assets to operate properly. Therefore, since it involves the specification of many information, the ability to model resources (f13) is a feature to be provided by RE languages [10].

A hazard is a state of the system that could ultimately lead to an accident that may result in a loss in human life [10]. Hence, the visualization of the impact level of an accident (f14), and the hazards associated as well as safety requirements should be supported by RE languages.

The support of textual description (f15) is essential for stakeholders that do not understand the graphical representation such as end user. Accordingly, a RE language may support it natively or be complemented with external artifacts. Scenarios or the Use Case Description are a popular choice to represent requirements and are adopted mainly in domain-independent approaches [8].

## 6. COMPARISON OF GORE LANGUAGES

An elaborated requirements engineering approach is crucial in the development of SCS in order to meet time, cost, and quality goals in SCS development [27].

Accordingly, safety concerns should be considered early in the development process, especially in the RE phase [1][8]. Considering this need, we evaluated the ability of four GORE languages to support a set of fundamental features described in Section 5.

In this paper, we characterized some GORE languages with respect to their capabilities against the proposed set of features for early specification requirements specification. We evaluated the languages using a three-grade scale score (Yes/Partially/No) as presented in Table 1. In this table, Y (yes) indicates that a given feature is supported, P means that a feature is Partially supported; N indicates that a feature is currently Not supported. The results are presented in Table 4.

**Table 4: Comparison of GORE modeling languages.**

| | Feature | i* | KAOS | GRL | NFR Framework |
|---|---|---|---|---|---|
| 1 | Modeling of accidents | N | P (Obstacle) | N | N |
| 2 | Modeling of hazards | N | P (Obstacle) | N | N |
| 3 | Modeling of causes of hazards | N | P (Sub-obstacles) | N | N |
| 4 | Modeling of environmental conditions | N | Y (Trigger conditions) | N | N |
| 5 | Modeling of functional safety requirements | Y (Tasks) | | | Y (Operationalizations) |
| 6 | Representation of constraints | Y (Contribution Links) | | | |
| 7 | Representation of obstacles | N | Y (Obstacle) | N | N |
| 8 | Representation of pre and post conditions | N | Y (pair Precondition, PostCondition) | N | N |
| 9 | Allow to represent the relationships among hazards, their causes, the environmental conditions and the functional safety requirements in a graphical form | N | N | N | N |
| 10 | Ability to specify how a particular event affects system safety | Y (Sofgoals and Contribution Links) | | | |
| 11 | Ability to specify the criticality level of safety-critical elements or the element's contributions to failure conditions | N | N | N | Y (Priority "!" symbol in softgoals) |
| 12 | Model and reasoning of safety strategies | Y (Sofgoals and Contribution Links) | | | Y (Operationalizations and Contribution Links) |
| 13 | Ability to model resources | Y (Resource Element) | | | Y (operationalizations) |
| 14 | Accident impact level | N | N | N | N |
| 15 | Support of textual description of safety requirements | N | N | N | N |

In summary, all surveyed approaches lack explicit modeling constructs to express how hazards can occur in the system, the accidents, their impact and how they can mitigated.

KAOS is a multiparadigm framework that allows to combine different levels of expression and reasoning: semi-formal for modeling and structuring goals, qualitative for selection among the alternatives, and formal, when needed, for more accurate reasoning [13].

Thus, the KAOS language combines semantic nets for conceptual modeling of goals, assumptions, agents, objects, and operations in the system, and linear-time temporal logic for the specification of goals and objects, as well as state-base

specifications for operations [13]. Since this language is composed by different models and have more concepts than other languages, KAOS better supports some features in relation to the other languages such as *Modeling of accidents (f1), Modeling of hazards (f2), Modeling of causes of hazards (f3), Modeling of environmental conditions* (f4), and *Representation of pre and post conditions* (f8).

KAOS was the only GORE language present in the SLR we conducted about the integration between RE and safety analysis [8]. The work found used the KAOS language to propose new tactics for elaborating system safety goals across a composite system.

i* (istar) is the second language in the ranking. The i* modeling language is basis for Tropos [22], a requirements-driven agent-oriented development methodology. i* is capable of representing the dependencies and relationships among actors in socio technical systems [23]. The features not supported by KAOS are either not supported by i*.

GRL has its roots in i* and the NFR Framework [25]. This language takes into account that not all high-level goals and non-functional requirements are equally important to the stakeholder. Therefore, an importance attribute (quantitative or qualitative) may be specified for intentional elements inside actors, which is used when evaluating strategies for the goal model [26]. i* and GRL have similar coverage.

Although NFR concentrates on systematically model and refine non-functional requirements and to expose positive and negative influences of different alternatives on these requirements, this language is the least appropriate language to specify the requirements of safety-critical systems. It does not have constructs to model accidents, hazards, their causes, environmental conditions, obstacles, and pre and post conditions.

NFR has the purpose of analyzing non-functional requirements and the elements that contribute or not to their satisfaction. Therefore, it does not have the ability to model resources (f13) among other features demonstrated in Table 4 and previously discussed.

## 7. THREATS TO VALIDITY

The threats to validity of Wohlin classification [28] considers four levels: construct, conclusion (reliability in qualitative research), internal, and external validity. We will discuss only the threats to validity we consider relevant in this work.

Construct validity threats were minimized with the properly definition of the safety-related concepts, presented in Section 4, used in this work. Hence, the safety constructs can be interpreted in the same way by the authors and readers.

We have evaluated four GORE languages and we were not able to find extensions for the languages related with safety. Considering that the GORE languages do not support safety, we are working on such extension.

Moreover, in this paper, we did not consider extensions to security or other domains that are too specific and do not cover the concepts and needs of SCS.

The extensions to security are not proposed to solve the safety problem, although they have some similar constructs, they are very different areas. However, security extensions may have the ability to (partially) support an early safety requirements specification, hence, we plan to increase the completeness of presented comparison in future studies.

One may argue that the assessment of the languages against the proposed set of criteria may be subjective or error prone. We have tried to mitigate this threat by trying to define and using clear features (see Section 5), and by having at least two people evaluating them. When we got disagreements, we checked by reading again the papers adopted for each language (Table 2). As in the work of Horkoff et al. [21], we have opted not to collect a formal method of agreement such as Kappa measure.

It is important to note that the authors have experience in goal modeling. The first author has experience more than five years in RE, the second have more than 20 years of experience in GORE with many publications with i*, the third author have more than 10 years of experience in RE, embedded systems and safety engineering, the fourth more than 15 years in RE, software engineering and safety engineering years, and the fifth more than 10 years in RE and GORE. This helps to increase our confidence in the results obtained.

## 8. RELATED WORK

Research regarding the improvement of the specification of Safety-Critical Systems (SCS), Self-Adaptive Systems (SAS) [40] and Systems-of-Systems (SoS) [41] with safety concerns has been an important concern in the literature.

In the systematic literature review [8] previously conducted, we extracted various pieces of safety-related information (data, concepts, knowledge, facts) used by the approaches that integrate requirements and safety engineering to document the safety concerns during the specification of SCS.

Another research line is the proposal of shared models with safety characteristics [5][42] or system engineering best practices [43]. In this context, UML [31][10] and SysML [7][38] profiles were proposed as well as safety analysis tools have been developed [44]. However, many of these studies do not aim to follow safety standards (or do it partially) and it is unclear whether these approaches can contest with the complexity of the large-scale development of software-intensive systems, taking inter-departmental and multi-disciplinary aspects into account [45].

Catalogs in the NFR framework specify the solutions to achieve a NFR like operationalizations. Although they are a good idea to represent solutions for common problems in safety, and may be proposed in future works, they do not provide explicit representation of the information that should be specified during the RE process. Moreover, there are no catalogues to safety, and the strategies to mitigate hazards and accidents are very much domain-dependent. Hence, it is out of scope to propose it.

# 9. CONCLUSIONS

The development of SCS comprehends the elicitation and specification of many information about hazards, their causes, environmental conditions, safety (functional) requirements and the relationships among them. Therefore, there is a tendency of considering these safety concerns early in the development process, especially in RE stage. However, currently, there is no consensus on the features an RE language must provide to support the description of such systems. This paper is a first step towards this direction.

Our study was motivated by three research questions, the first one regarding the conceptual foundation for safety requirements specification in RE process, the second one about the main features that requirements languages should support in terms of specification of safety-critical systems; and the third one about the similarities and differences among GORE languages support for the specification of safety-critical systems.

We proposed a set of 15 features that requirements languages should provide to an early safety requirements specification. This set is a novel contribution that can be used as criteria to guide the selection of requirements languages for specifying SCS. Moreover, we characterized the support of some goal-oriented languages in this specification. We evaluated four GORE languages (i*, KAOS, GRL, NFR-Framework) against this set of features.

As a result from our evaluation, we observe that none of these languages fully support the specification of SCS considering the set of proposed features. Nevertheless, KAOS and i* appears to be promising languages in which extensions could be proposed to support the features.

An extension of a GORE language can be used to represent the results of model-based safety analysis techniques. We believe that an extension is necessary and it is under development.

In the next section, we suggest further research on specification of safety-critical systems.

## 9.1. Further research

The results of our comparative analysis pointed out that the evaluated languages do not have constructs to modeling safety-related aspects. Therefore, this study has generated some research directions:

- How is the support of non-GORE languages such as UML to describe SCS requirements?
- In what extent do extensions to security cover the concepts and needs of SCS?
- How can we evaluate the completeness of the proposed set of features?
- How can we adapt the evaluated languages to support the features?

# ACKNOWLEDGMENTS

This work was partially supported by FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

# REFERENCES

[1] Luiz Eduardo G. Martins; Tony Gorschek. Requirements engineering for safety-critical systems: A systematic literature review. Information and Software Technology, v. 75, pp. 71-89, 2016. DOI: http://dx.doi.org/10.1016/j.infsof.2016.04.002

[2] Nancy Leveson. System safety and computers. Addison Wesley, 1995.

[3] Luiz Eduardo G. Martins; Tiago de Oliveira. A case study using a protocol to derive safety functional requirements from fault tree analysis. In: Requirements Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, 2014. pp. 412-419. DOI: http://dx.doi.org/10.1109/RE.2014.6912292

[4] Luiz Eduardo G. Martins, Hanniere de Faria, Lucas Vecchete, Tatiana Cunha, Tiago de Oliveira, Dulce E. Casarini, and Juliana Almada Colucci. Development of a Low-Cost Insulin Infusion Pump: Lessons Learned from an Industry Case. In: Computer-Based Medical Systems (CBMS), 2015 IEEE 28th International Symposium on. IEEE, 2015. pp. 338-343. DOI: http://dx.doi.org/10.1109/CBMS.2015.14

[5] Nancy Leveson. An approach to designing safe embedded software. In: International Workshop on Embedded Software. Springer Berlin Heidelberg, 2002. pp. 15-29. DOI: 10.1007/3-540-45828-X_2

[6] Ben Swarup Medikonda;,Seetha Ramaiah Panchumarthy. A framework for software safety in safety-critical systems. ACM SIGSOFT Software Engineering Notes, v. 34, n. 2, pp. 1-9, 2009. DOI: 10.1145/1507195.1507207

[7] Sven Scholz; Kleanthis Thramboulidis. Integration of model-based engineering with system safety analysis. International Journal of Industrial and Systems Engineering, v. 15, n. 2, pp. 193-215, 2013. DOI: 10.1504/IJISE.2013.056096

[8] Jéssyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, and Tony Gorschek. Integration between requirements engineering and safety analysis: A systematic literature review. Journal of Systems and Software, v. 125, pp. 68-92, 2017. DOI: https://doi.org/10.1016/j.jss.2016.11.031

[9] Kleanthis Thramboulidis; Sven Scholz. Integrating the 3+ 1 SysML view model with safety engineering. In: Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on. IEEE, 2010. pp. 1-8. DOI: 10.1109/ETFA.2010.5641353

[10] Gregory Zoughbi; Lionel Briand; Yvan Labiche. Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile. Software & Systems Modeling, v. 10, n. 3, pp. 337-367, 2011. DOI: 10.1007/s10270-010-0164-x

[11] ISO, International Organization for Standardization. 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems, International Electrotechnical Commission.

[12] RTCA: Software Considerations in Airbone Systems and Equipment Certification. Radio Technical Commission for Aeronautics (RTCA), European Organization for Civil Aviation Electronics (EUROCAE), Standard Document no. DO-178B/ED-12B, December 1992

[13] Alexei Lapouchnian. Goal-oriented requirements engineering: An overview of the current research. University of Toronto, pp. 32, 2005.

[14] Annie Anton. Goal-based requirements analysis. In: Requirements Engineering, 1996., Proceedings of the Second International Conference on. IEEE, 1996. p. 136-144. DOI: 10.1109/ICRE.1996.491438

[15] Evangelia Kavakli; Pericles Loucopoulos. Goal driven requirements engineering: evaluation of current methods. In: Proceedings of the 8th CAiSE/IFIP8. 2003. pp. 16-17.

[16] Jasen Markovski; J. M. Van de Mortel-Fronczak. Modeling for safety in a synthesis-centric systems engineering framework. In: International Conference on Computer Safety, Reliability, and

Security. Springer Berlin Heidelberg, 2012. pp. 36-49. DOI: 10.1007/978-3-642-33675-1_4

[17] Military Standard. System safety program requirements. MIL-STD-882C, US Department of Defense, USA, 1993.

[18] U. S. Dod MIL-STD-882E, Department of Defense Standard Practice System Safety. US Department of Defense, 2012.

[19] Jim Whitehead. Collaboration in Software Engineering: A Roadmap. In Future of Software Engineering (FOSE '07). IEEE Computer Society, Washington, DC, USA, 2207, pp. 214-225. DOI=http://dx.doi.org/10.1109/FOSE.2007.4

[20] Luiz Eduardo G. Martins; Tony Gorschek. Requirements Engineering for Safety-Critical Systems: Overview and Challenges. Accepted for publication. In: IEEE Software, 2017. For a copy: legmartins@unifesp.br.

[21] Jennifer Horkoff, Tong Li, Feng-Lin Li, Mattia Salnitri, Evellin Cardoso, Paolo Giorgini, John Mylopoulos, and Joao Pimentel. Taking goal models downstream: a systematic roadmap. In: Eighth International Conference on Research Challenges in Information Science (RCIS), 2014. pp. 1-12. DOI: 10.1109/RCIS.2014.6861036

[22] Jaelson Castro; Manuel Kolp; John Mylopoulos. A requirements-driven development methodology. In: International Conference on Advanced Information Systems Engineering. Springer Berlin Heidelberg, 2001. pp. 108-123. DOI: 10.1007/3-540-45341-5_8

[23] S. K Eric. Social modeling for requirements engineering. Mit Press, 2011.

[24] Anne Dardenne; Axel Van Lamsweerde; Stephen Fickas. Goal-directed requirements acquisition. Science of computer programming, v. 20, n. 1-2, pp. 3-50, 1993.

[25] John Mylopoulos; Lawrence Chung; Brian Nixon. Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on software engineering, v. 18, n. 6, pp. 483-497, 1992.

[26] Daniel Amyot; Gunter Mussbacher. Development of Telecommunications Standards and Services with the User Requirements Notation. In: Workshop on ITU System Design Languages, 2008.

[27] John Hatcliff, Alan Wassyng, Tim Kelly, Cyrille Comar, and Paul Jones. Certifiably safe software-dependent systems: challenges and directions. In: Proceedings of the on Future of Software Engineering. ACM, 2014. pp. 182-200. DOI: 10.1145/2593882.2593895

[28] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. Experimentation in software engineering. Springer Science & Business Media, 2012.

[29] Nancy Leveson. System safety and computers. Addison Wesley, 1995.

[30] Alan Simpson; Joanne Stoker. Will it be Safe?—An Approach to Engineering Safety Requirements. In: Components of System Safety. Springer London, 2002. pp. 140-164. DOI: 10.1007/978-1-4471-0173-4_9

[31] Javier Fernández Briones, Miguel Ángel De Miguel, Juan Pedro Silva, and Alejandro Alonso. Application of safety analyses in model driven development. Software Technologies for Embedded and Ubiquitous Systems, p. 93-104, 2007. DOI: 10.1007/978-3-540-75664-4_10

[32] Samuel Fricker; Tony Gorschek; Martin Glinz. Goal-oriented requirements communication in new product development. In: Second International Workshop on Software Product Management, IWSPM'08, 2008. pp. 27-34. DOI: 10.1109/IWSPM.2008.2

[33] Ernst SIKORA; Bastian TENBERGEN; Klaus POHL. Industry needs and research directions in requirements engineering for embedded systems. In: Requirements Engineering, v. 17, n. 1, pp. 57-78, 2012. DOI: 10.1007/s00766-011-0144-x

[34] Martin GLINZ; Samuel A. Fricker. On shared understanding in software engineering: an essay. Computer Science-Research and Development, v. 30, n. 3-4, pp. 363-376, 2015. DOI: 10.1007/s00450-014-0256-x

[35] Barbara Paech,; Jorg Dorr; Mathias Koehler. Improving requirements engineering communication in multiproject environments. IEEE software, v. 22, n. 1, pp. 40-47, 2005. DOI: 10.1109/MS.2005.10

[36] Sultan Aljahdali; Jameela Bano; Nisar Hundewale. Goal Oriented Requirements Engineering-A Review. In: 24th International Conference on Computer Applications in Industry and Engineering, Honolulu, Hawaii, USA, CAINE. 2011. pp. 16-18.

[37] Sadaf Mustafiz; Jörg Kienzle. DREP: A requirements engineering process for dependable reactive systems. In: Methods, Models and Tools for Fault Tolerance. Springer Berlin Heidelberg, 2009. p. 220-250. DOI: 10.1007/978-3-642-00867-2_11

[38] Geoffrey Biggs; Takeshi Sakamoto; Tetsuo Kotoku. A profile and tool for modelling safety information with design information in SysML. Software & Systems Modeling, v. 15, n. 1, p. 147-178, 2016. DOI: 10.1007/s10270-014-0400-x

[39] Samuel Fricker, Tony Gorschek, Carl Byman, Armin Schmidle. Handshaking with implementation proposals: Negotiating requirements understanding. IEEE software, v. 27, n. 2, p. 72, 2010.

[40] Monique Soares; Jéssyka Vilela; Gabriela Guedes; Carla Silva; Jaelson Castro. Core Ontology to Aid the Goal Oriented Specification for Self-Adaptive Systems. In: New Advances in Information Systems and Technologies. Springer International Publishing, 2016. pp. 609-618. DOI: 10.1007/978-3-319-31232-3_57

[41] Milena Guessi; Everton Cavalcante; Lucas Oliveira. Characterizing architecture description languages for software-intensive systems-of-systems. In: Proceedings of the third international workshop on software engineering for systems-of-systems. IEEE Press, 2015. pp. 12-18.

[42] Rajiv Murali; Andrew Ireland; Gudmund Grov. A rigorous approach to combining use case modelling and accident scenarios. In: NASA Formal Methods Symposium. Springer International Publishing, 2015. pp. 263-278. DOI: 10.1007/978-3-319-17524-9_19

[43] Romaric Guillerm; Hamid Demmou; Nabil Sadou. Information model for model driven safety requirements management of complex systems. In: Complex Systems Design & Management. Springer Berlin Heidelberg, 2010. pp. 99-111. DOI: 10.1007/978-3-642-15654-0_7

[44] Vivek Ratan, Kurt Partridge, Jon Reese, and Nancy Leveson. Safety analysis tools for requirements specifications. In: Computer Assurance, 1996. COMPASS'96, Systems Integrity. Software Safety. Process Security. Proceedings of the Eleventh Annual Conference on. IEEE, 1996. pp. 149-160. DOI: 10.1109/CMPASS.1996.507883

[45] Joakim PERNSTÅL, Tony Gorschek, Robert Feldt, and Dan Florén. Requirements communication and balancing in large-scale software-intensive product development. Information and Software Technology, v. 67, pp. 44-64, 2015. https://doi.org/10.1016/j.infsof.2015.06.007

[46] Jennifer Horkoff; Eric Yu. Analyzing goal models: different approaches and how to choose among them. In: Proceedings of the 2011 ACM Symposium on Applied Computing. ACM, 2011. pp. 675-682. DOI: 10.1145/1982185.1982334

[47] Nancy Leveson. Engineering a Safer World: Systems Thinking Applied to Safety. Mit Press, 2011

[48] Nancy Leveson. Safeware: System Safety and Computers. ACM, 1995.

[49] Paulo Lima, Jéssyka Vilela, Enyo Gonçalves, João Pimentel, Ana Holanda, Jaelson Castro, Fernanda Alencar, Maria Lencastre. Scalability of iStar: a Systematic Mapping Study. In: Proceeding of Workshop of Engenharia de Requisitos (WER), 2016.