PRAOS: A Process to Support KAOS Extensions

Enyo Gonçalves[1], João Araujo[2], Leandro Monte[1] and Marcos Oliveira[1]

[1]Universidade Federal do Ceará, Brazil
[2]Universidade Nova de Lisboa, Portugal

Abstract

Goal-Oriented Requirements Engineering (GORE) supports the development team in identifying the requirements that the system must fulfil. Commonly applied during the initial stages of requirements gathering, it focuses on identifying the system's goals, presenting their decomposition as a means of offering alternatives to satisfy them. Knowledge Acquisition in Automated Specification (KAOS) is a GORE approach that comprises a method, a software environment, and a modeling language. Modelling languages can be adaptable to various domains/application areas where the software will be developed. This way, extensions are proposed to adapt the modelling to the desired scenarios. These adaptations are referred to as extensions. KAOS has been extended to various areas, including security, adaptive systems, and aspects, among others. The creation of new KAOS extensions has been growing. It is expected to continue in the coming years, as it is necessary to adapt languages to the various existing contexts and those that emerge with the constant evolution in software development. Creating an extension is a complex task with inherent challenges, such as maintaining consistency between the developed and existing extensions. Given these facts, we recognise the need to support the creation of KAOS language extensions. This study aims to support the systematic creation of new KAOS extensions through a systematic process. The proposed process was used to create a new KAOS extension to represent accessibility concepts, which proved valid for this purpose. Finally, the PRAOS process was evaluated by KAOS extension specialists through a qualitative study.

1. Introduction

Goal-Oriented Modelling Languages visually represent the system to be developed and are used in the early stages of development to elicit and specify application requirements. Errors in Requirements Engineering (RE) can lead to project failure if they deviate from expected outcomes [13]. Goal-oriented modelling languages help identify, structure, and validate requirements, thereby reducing RE errors [13]. According to Matulevicius and Heymans [15], graphical representations in modelling languages improve system understanding among stakeholders and developers, ensuring accurate project direction.

KAOS [5] is a goal-oriented requirements modelling language used in software development. Extending a modelling language involves adding new constructs to expand its representation capacity. Over the years, several KAOS extensions have been proposed; however, they were developed non-systematically, resulting in issues such as inconsistency and a lack of CASE tool support. A Systematic Literature Review (SLR) of KAOS extensions [8] identified 50 KAOS extensions; we can mention [18] as an example of a KAOS extension identified by this SLR.

The evolution of software development, driven by new technologies and paradigms, has necessitated the extension of KAOS to ensure a faithful and precise specification of requirements, effectively bridging this gap. These extensions aim to enable a more precise specification of systems to be developed. Thus, we argue that KAOS has been extended and will be continue to be extended. However, these extensions have been proposed in an ad hoc way.

This work supports the systematic creation of KAOS extensions through a systematic process. Processes provide consistency and structure, essential for successful project execution and knowledge transfer [16]. Our process is derived from PRISE [9], with several key modifications. Firstly, we incorporated the use of LLMs in several sub-processes. Secondly, we adapted artefacts and tasks to align

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France
enyo@ufc.br (E. Gonçalves); joao.araujo@fct.unl.pt (J. Araujo); leandromonte@alu.ufc.br (L. Monte); marcos.oliveira@ufc.br (M. Oliveira)

with KAOS specifics, including: a List of Experts in KAOS extensions; a Catalog of KAOS extensions and their constructs; Guidelines; the KAOS metamodel; a Checklist for Completeness, Consistency, and Conflicts; an extension specification template; and a List of tools for task descriptions in sub-processes to adapt existing tools. Third, we ensured the alignment of tasks and artefacts with KAOS concepts. Fourth, we integrated a catalog of KAOS extensions [4] to ease the search for KAOS extensions and their constructs. Last, we proposed KAOS4Ext, a tool to support KAOS modelling and extension.

This paper is organized as follows: Section 2 presents the background, Section 3 shows related work, Section 4 details our process and its validation, Section 5 evaluates the process through interviews, and Section 6 concludes and suggests future work.

2. BACKGROUND

2.1. Modelling Languages Development and Extension

Model-based engineering (MBE) can be defined as a process in which software models play an important role but are not necessarily critical artefacts of the development [1]. An example of the MBE is when models document the system, and automatic code generation is not involved. Models are first-class artefacts of the software development process in Model-Driven Engineering (MDE) [1], from which part of the application is generated. Researchers working in the modelling languages area have focused on the Abstraction Challenge: What kind of modelling constructs and the underlying foundation are needed to support the development of domain constructs that are considered first-class modelling elements in a language? [1].

In this context, modelling language is defined by its abstract syntax (i.e., through a metamodel and well-formedness rules/semantics) and concrete syntax. According to Kelly and Tolvanen [11], the modelling language constructs should be formalized, and it is best specified by defining its metamodel, since they emphasize that metamodelling simply means modelling your language: mapping your application area concepts to various language elements, their properties, and their connections, specified as links and the roles that elements play in them.

We typically identify various domain rules, constraints, and consistency requirements that a language should adhere to. These rules obviously need to be defined too. Having rules in the language provides several benefits, including early error prevention, guidance towards preferable design patterns, checks for completeness by informing about missing parts, minimization of modeling work effort through conventions and default values, and consistency of specifications [11].

Modelling language designers use abstract syntax and well-formedness rules as the starting point for concrete syntax definition [1]. Thus, a set of models and their graphical and textual elements should be proposed for a modelling language to be usable. According to France and Rumpe [7], modelling tools can be essential in reducing the accidental complexities associated with understanding and using modelling languages.

Brambilla et al. [1] define extending a modelling language as adding new constructs, modifying or removing existing constructs, based on the needs of the domain in which the extension is being applied. Besides, a new extension provides greater flexibility and the ability to capture and communicate the essence of complete systems.

2.2. KAOS and KAOS Extensions

KAOS [5] is a Goal-Oriented Requirements Engineering (GORE) modelling language that allows the modelling of functional and non-functional requirements by using a combination of four models: goal model, responsibility model, object model, and operation model [5]. These models are based on objectives, requirements, agents, expectations, obstacles, domain ownership, operations, entities, events, and links between these concepts. They are detailed below:

- Goal Model - The goal model is considered the basis and starting point. It represents a set of interrelated objective diagrams used to address a problem. The primary concept behind this

approach is to represent system requirements as business goals and objectives, thereby focusing on achieving these business objectives. Goals are typically comprised of both functional and non-functional requirements that must be incorporated into the system being developed.

- Responsibility Model - The KAOS responsibility model is a compilation of derived responsibility diagrams. It involves agents, which can be human beings or automated components, concerned with achieving goals/requirements. The assignment of agents to fulfill the specific objective is done according to the goal model. Goals are always assigned to multiple agents. However, whenever there is a single agent response to the objective, there is no room for any further objective refinement, and this difference provides the analyst with a criterion to stop refining objectives into sub-goals. A responsibility diagram describes, for each agent, the requirements and expectations for which they are responsible or assigned.
- Object model - The object model is related to the application domain binding and establishing restrictions in the system. Objects can be categorized as entities, agents, and associations, where entities describe and translate the object's state but do not perform operations; agents are concerned with the execution of operations, while associations are entities that depend on the object and cannot perform operations.
- Operation model - The operation model represents all agents' behaviours. Behaviours are basically operations performed by agents. These operations are used to manipulate the objects described in the object model: they can create objects, cause object state transitions, or trigger other operations through events sent and received.

KAOS has been extended to represent new constructs, adapt existing ones, or remove them. From this need arises the creation of new KAOS extensions, which involve adapting an existing extension to new scenarios, such as new representations and specific functionalities tailored to the particular application domain.

A Systematic Literature Review (SLR) of KAOS extensions [8] identified fifty KAOS extensions and a total of 221 new constructs. Figure 1 shows an example of using a KAOS extension for modelling custom adaptive systems [18]. We can identify elements not part of the standard KAOS syntax, such as softgoal.

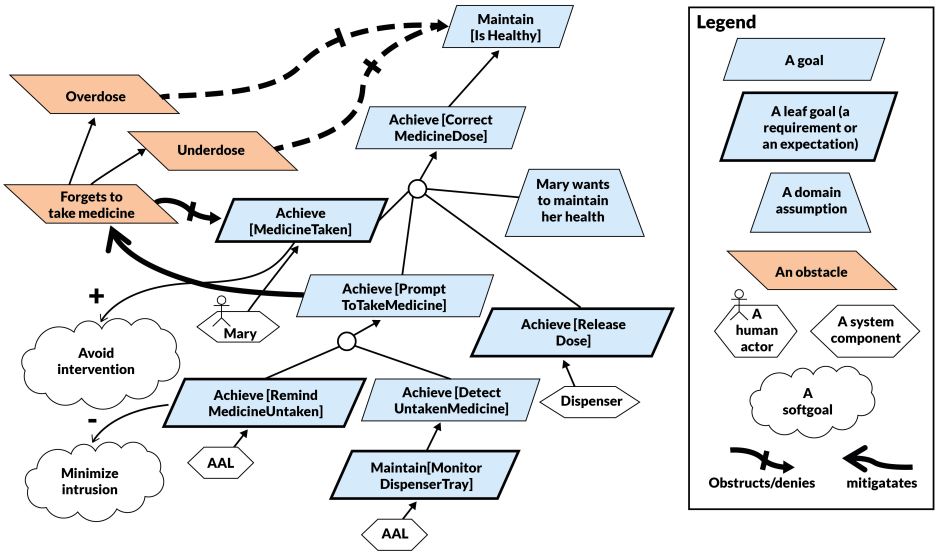

Figure 1: Example of a KAOS Extension to Model Adaptive Systems [18].

The first KAOS extension identified by [8] was proposed in 2006, and the last one was identified in 2024 (final year considered by this SLR). The SLR of KAOS extensions [8] indicates that 2011 was the year with the highest number of KAOS extensions proposed (6 papers) and a mean of 2.63 extensions proposed

per year. The main application areas for KAOS extensions are organizational/business processes (12 extensions), adaptive systems (9 extensions), and security/privacy/vulnerability (7 extensions).

An analysis of the extensions revealed that almost half of the papers (48%) provide complete definitions for all introduced concepts, while 44% offer partial definitions, and 8% do not present any definition. Regarding syntax level, most extensions (72%) focused only on concrete syntax , 24% addressed both abstract and concrete syntax, and 4% only abstract syntax. Furthermore, most abstract syntax extensions (92%) are conservative, maintaining all KAOS constructs. As for new constructs, 62.44% were applied to nodes, and 37.56% to links. A crucial point is that only 35% of the extensions have CASE tool support, which may limit their adoption and practical application. No standardized extension mechanisms were identified for KAOS. These data highlighted the need to support the proposal for the next KAOS extensions.

Gonçalves et al. [9] supported the creation of new extensions of the iStar modelling language through a process to support iStar extensions, called PRISE. The process is based on well-defined sub-processes, tasks, and artefacts. PRISE supports the iStar and its variations such as GRL, TROPOS and Secure TROPOS. PRISE [9] was the start point to PRAOS creation. It provided guidance and was adapted to KAOS specificities, such as the KAOS metamodel and constructs, as well as specific support tools, including the catalogue of KAOS extensions[4].

Fowler [6] describes how the UML (Unified Modelling Language) presents extension mechanisms, thus enabling its adaptation to the specific needs of designers, such as adaptation to a new domain. The UML extension mechanisms are profiles, tagged values, stereotypes, and constraints. Textual stereotypes and tagged values (which are part of UML extension mechanisms) are frequently adopted by other modelling languages as a practical and recognized means to represent their own extensions. To address this, PRISE includes KAOS4Ext, a tool developed to support the modelling of KAOS and its extensions. This tool enables the creation of new constructs based on textual stereotypes and tagged values, while also supporting the definition of new graphical symbols, thereby providing practical support during Sub-Process 3 ('Develop KAOS Extension') of PRAOS.

Braun et al. [2] addressed the extensibility of business modelling languages, especially the Business Process Model and Notation (BPMN). A method was proposed to support the creation of extensions to business modelling languages, distinguishing itself from other methods by justifying the need for an extension at a semantic level, guided by ontology. This method consists of three steps: Domain Analysis, Extension Preparation, and Extension Metamodel Design. The last step expands and adapts the syntax extension method in BPMN previously proposed by Stroppi et al. [17]. While focused on business processes, this work highlights the relevance of a structured, multi-step process for language extension, similar to PRAOS. Both approaches recognize the need to systematically analyze the domain, define new concepts, and evolve the language's meta-model. However, PRAOS specifically tailors this systematic approach to the unique characteristics and challenges of extending Goal-Oriented Requirements Engineering languages, particularly KAOS, building upon its specific metamodel and community guidelines.

3. Description of the PRAOS Process

The proposal for a process to support the creation of new KAOS extensions is based on the PRISE process by Gonçalves et al. [9]. PRISE supports extensions in the iStar language, which is goal-oriented and serves as the basis for developing PRAOS, a process adapted for KAOS. The analysis of PRISE is detailed below.

3.1. Analysis of PRISE

This section will describe the steps and central adaptations made to PRISE to adjust it to the creation of KAOS extensions, resulting in a new version called PRAOS (PRocess to support kAOS extensions).

The adaptation process followed these steps: First, we created a spreadsheet containing all PRISE sub-processes, tasks, artefacts, and gateways. Each element of this spreadsheet was classified as

"remove", "keep", or "keep adapted". When necessary, the adaptations were detailed in the spreadsheet itself. This analysis was supervised by a researcher with experience in extensions and required five review/correction cycles to complete a spreadsheet[1].

After identifying the changes in the spreadsheet, these changes were implemented in Bizagi's PRISE project. PRISE was initially modelled in BPMN (Business Process Modelling and Notation), and we decided to keep the PRAOS modelling in BPMN as well, given that this language is widely used for modelling business processes. This stage was also supervised by a researcher with experience in extensions, with five review/correction cycles of the modelled process.

The changes can be categorized into two classes: more general and simple changes, which affect a set of parts of the process (such as changing the name from PRISE to PRAOS), and more specific changes (such as replacing the list of iStar constructs with KAOS constructs in the checklist for verification of completeness, consistency, and conflicts). We made several general changes to the PRISE process to adapt it to KAOS, resulting in significant modifications to the sub-processes. Most of these changes involved replacing references to iStar with KAOS, which essentially implied changing the terminology and concepts associated with modelling. Therefore, most changes consisted of updating the terms and names related to the new modelling language. Initially, we could reuse 34 PRISE elements (tasks/artifacts/gateways) without the need for significant adaptations since these elements were already organized in a general way, including gateways and initial and final states. Next, we analysed the PRISE structures that were compatible with PRAOS but required minor adjustments, such as replacing terms related to the KAOS language. These adjustments represented most of the modifications made from PRISE, accounting for 96 changes considered simple. Finally, we addressed 27 more specific adaptations. We can mention as examples of these adaptations, the change of the iStar metamodel to the KAOS metamodel, the development of a catalogue of KAOS extensions, a modelling tool to support KAOS extensions, the list of the experts in KAOS extensions, the adaptation of iStar extensions guidelines to KAOS, and the introduction of LLMs usage in some steps of the process.

3.2. Describing PRAOS

This Section details PRAOS[2]. Additionally, we presented parts of a new KAOS extension[3]. The proposed extension focuses on representing accessibility concepts in KAOS modeling.

PRAOS consists of five subprocesses and one task that occurs in parallel to three subprocesses. Figure 2 presents the primary process. It is based on a set of nine (9) guidelines adapted from a previous study [9]. These guidelines were evaluated by experts in KAOS extensions(Section V).

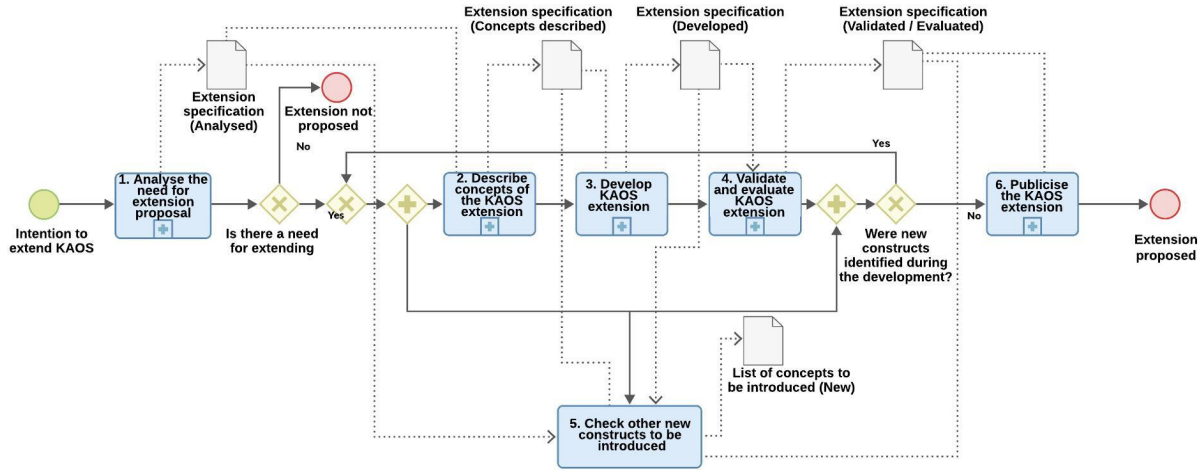

Figure 2: PRAOS Main Process

[1]http://bit.ly/44WSYEb
[2]https://bit.ly/PRAOS-EN
[3]The complete extension specification is available at https://bit.ly/ext-spec-PRAOS

In the following, we describe the adapted and recommended procedures for creating a KAOS extension proposal.

G1: Preserving the original syntax of the KAOS language is essential. It is recommended that all nodes and links of the original syntax be maintained in the extension. In other words, extensions that remove all standard constructs, those that are not conservative, are generally discouraged and considered examples not to be followed.

G2: Extensions must be consistent, complete, and conflict-free, following a defined process or method for their creation. Currently, the creation of extensions for KAOS has been carried out haphazardly without centralizing existing information about the modeling language.

G3: Conduct a systematic review of the literature to identify all existing extensions and assess the absolute need to create a new extension or take advantage of an existing one. It is essential to consider including a KAOS expert in the development process and modeling several systems in the desired area to gain a more comprehensive understanding of the domain that requires the extension.

G4: Describe the extension concepts clearly and objectively, seeking to minimize the possibility of divergent interpretations as much as possible.

G5: Consider proposing both the abstract and concrete syntax of the extension, as they can complement each other.

G6: Seek to maintain a clear and direct reformulation, focusing on consistency between the two syntaxes.

G7: Establish a relationship between the constructs inserted in the extension and those existing in the original KAOS language.

G8: Seek to define new extensions with the smallest possible number of new constructs, making the most of existing constructs and creating only the minimum necessary to adapt to the new domain.

G9: Try to create simple representations that can be understood without the need for specific tools, thus ensuring ease of use.

The following sections provide a detailed description of each of these sub-processes and their related artifacts.

3.3. Analyse the Need for Extension Proposal (Sub-Process 1)

This subprocess aims to verify the need to propose a new KAOS extension and establish the initial requirements for such an extension. Artefacts used by subprocess 1: List of references and contacted researchers, List of concepts to be introduced, Modelling and observations, List of experts in KAOS extensions, List of related KAOS extensions and Extension specification (analysed). Figure 3 presents sub-process 1.

Initially, the rationale for the extension proposal is analysed, and the new concepts to be introduced are identified. It generates the Extension specification [Analysed]. Subsequently, a thorough study of the application area is conducted to identify specific requirements and gaps that the standard KAOS cannot address. This involves a literature review and, if necessary, consultation with domain experts. A review is not required when it relates to the practical aspects of KAOS. Based on this analysis, the new concepts that the extension should encompass are identified. These concepts may relate to the application domain or more general aspects of KAOS.

To validate the need for the extension, five sequential tests are performed. The first test assesses whether the application domain exhibits problems that cannot be modelled using standard KAOS. Subsequent tests evaluate the feasibility of modelling the new concepts using existing KAOS and identify the limitations. Two tests involve consulting with experts in the domain and KAOS extensions. We suggest that when these experts are not available, LLMs (such as ChatGPT[4] and Gemini[5]) can be used to mitigate issues. Questions can be made to these LLMs about the list of new constructs and issues related to KAOS constructs. Task 1.10 is related to the identification of existing KAOS extensions;

[4]https://chatgpt.com/

[5]gemini.google.com

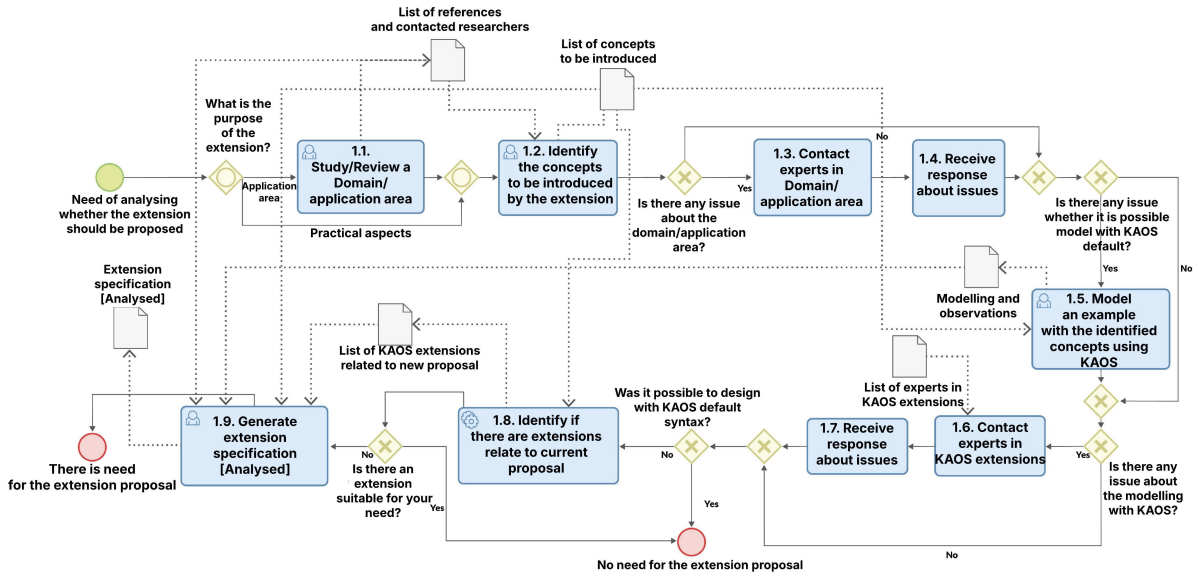

Figure 3: PRAOS Sub-process 1 Analyse the Need for Extension Proposal.

it can be performed based on the support of the catalogue of KAOS extensions[6] [4]. At the end of this process, a decision is made regarding the viability of the extension. If a genuine need is identified, the extension development process can start. Otherwise, the process is terminated.

According to W3C [19], system accessibility aims to create technologies that can be used by all types of people, regardless of their physical, sensory, or cognitive abilities. The inclusion of practices that guarantee system accessibility ensures that people with disabilities can access websites, software, and devices in the same way that people without disabilities can access them—ensuring digital inclusion and preventing people with disabilities from feeling isolated from the technological world.

After identifying the context and need to create the extension, a (non-systematic) literature review was carried out. Thus, based on these steps, the following concepts were identified: Accessible Agent, Accessibility Requirement, Accessible Operation, and Accessible Domain Property.

It was not necessary to consult domain experts or an LLM. We modeled an example of an accessible system using KAOS (See the complete extension specification) and identified the need to represent accessibility concepts with a specific syntax to highlight them in KAOS models. It was not necessary to consult KAOS experts, since the author is an expert in KAOS extensions. We did not find any related extension in the KAOS extension catalogue.

3.4. Describe Concepts of the KAOS Extension (Sub-Process 2)

This sub-process involves several tasks and artefacts, with the Extender and the Expert in KAOS extensions as key participants. The primary artefact generated by this sub-process is the "Extension Specification [Concepts Described]". Artefacts used by subprocess 2: Extension specification (concepts described); Modelling and observations; List of relations between KAOS original constructs and extension constructs; List of ideas to be introduced and list of concepts to be reused. Figure 4 presents sub-process 2.

The initial task involves searching and selecting reusable constructs. The Extender accesses the KAOS extensions catalogue to identify and choose the constructs previously identified in the "Extension Specification [Analysed]". Next, the process focuses on describing the extension's concepts in detail. This includes defining the list of constructs introduced or reused within the KAOS extension.

Subsequently, the Extender analyzes how to integrate these new extension constructs with the existing KAOS constructs. If any integration challenges arise, the Extender consults with experts in

[6]https://kaos-catalogue-d9e49.web.app/

KAOS extensions. The Expert in KAOS extensions then provides guidance and resolves the integration issues. LLMs can be helpful in this context, clarifying issues related to the meaning of constructs and their integration with KAOS default constructs.

Finally, the "Extension Specification [Concepts Described]" is generated by combining the "Extension Specification [Analysed]" with the artefacts produced in this sub-process: the "List of Constructs to be Reused" (including their definitions) and a list defining the relationships between extension and KAOS constructs. This sub-process concludes with a complete description of the KAOS extension concepts.

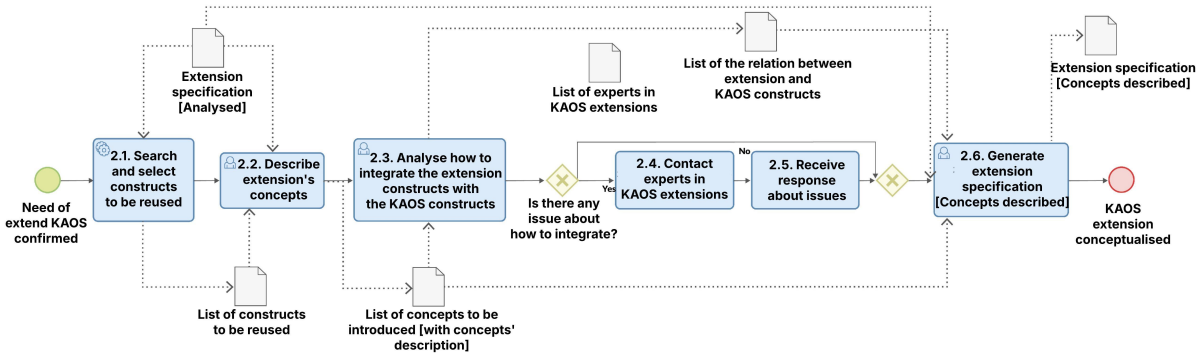

Figure 4: PRAOS Sub-process 2 Describe Concepts of the KAOS Extension.

3.5. Develop KAOS Extension (Sub-Process 3)

This sub-process, initiated upon confirmation of the need for extending KAOS in sub-process 1, focuses on defining the new concepts that will be introduced into the KAOS modelling language. The Extender is the primary actor in this phase. Before proceeding, the Extender should carefully review the guidelines provided by KAOS experts for developing extensions. Artefacts used in subprocess 3 include the KAOS metamodel, Extension metamodel, validation rules, Extension specification (Developed), and Checklist of problem verification. Figure 5 presents sub-process 3.

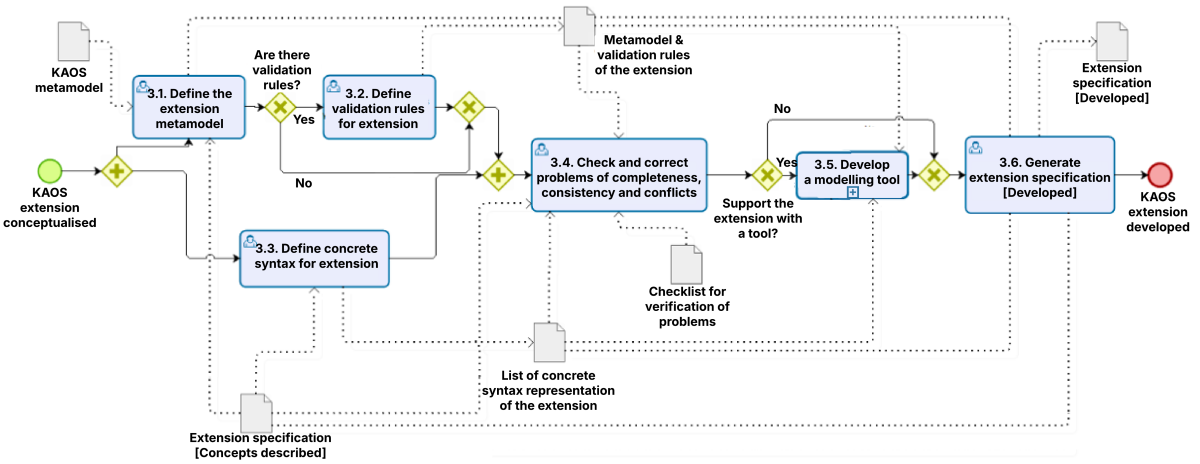

Figure 5: PRAOS Sub-process 3 Develop KAOS Extension

The first step involves defining the metamodel for the extension. This includes determining the new concepts, their attributes, and their relationships with existing KAOS constructs. The Extender should refer to existing KAOS metamodels, such as those presented by Lamsweerde [10] and Matulevicius et al. [13], as a starting point for incorporating these new extension constructs. The guidelines of the KAOS community, particularly G1 (Preserve the original KAOS syntax) and G7 (Establish relationships between extension constructs and existing KAOS constructs), should be strictly adhered to.

Next, the Extender defines validation rules for the extension. These rules address constraints that cannot be directly represented within the metamodel. For example, if certain combinations of constructs are not allowed, the Extender would define rules to enforce these constraints.

Concurrently, the Extender defines the concrete syntax for the extension. This involves determining the graphical representation of the new extension constructs. One approach is to experiment, as Caire et al. [3] did, to explore different representation options. The Extender should consider the suggestion that the first level of integration involves specializing in existing KAOS constructs, such as using textual markers as stereotypes. If this approach is not feasible, new graphical representations should be proposed. The impact of these new graphical representations on each of the four standard KAOS diagrams should be carefully considered. The Extender should strive to propose graphical representations that are distinct from those used in existing KAOS extensions, while also considering the reuse of existing KAOS constructs where appropriate.

We suggest using LLMs (such as ChatGPT and Gemini) to mitigate issues with the metamodel definition and concrete syntax. LLMs can contribute to clarifying the representation of a new element in the KAOS metamodel. LLMs can help indicate symbols to represent the new constructs.

Once the proposal for concrete syntaxes is finalized, a check is performed to ensure the completeness, consistency, and absence of conflicts within the extension. This analysis covers the completeness of the extension definition (including concepts, abstract syntax, and concrete syntax), consistency between concepts, abstract syntax, and concrete syntax, the absence of nodes and links that conflict with the standard KAOS syntax, the occurrence of conflicts between new constructs and existing KAOS constructs, and the correctness of the representation of standard KAOS constructs within the extension. The Extender should use a checklist to verify the occurrence of these potential problems and correct any issues identified.

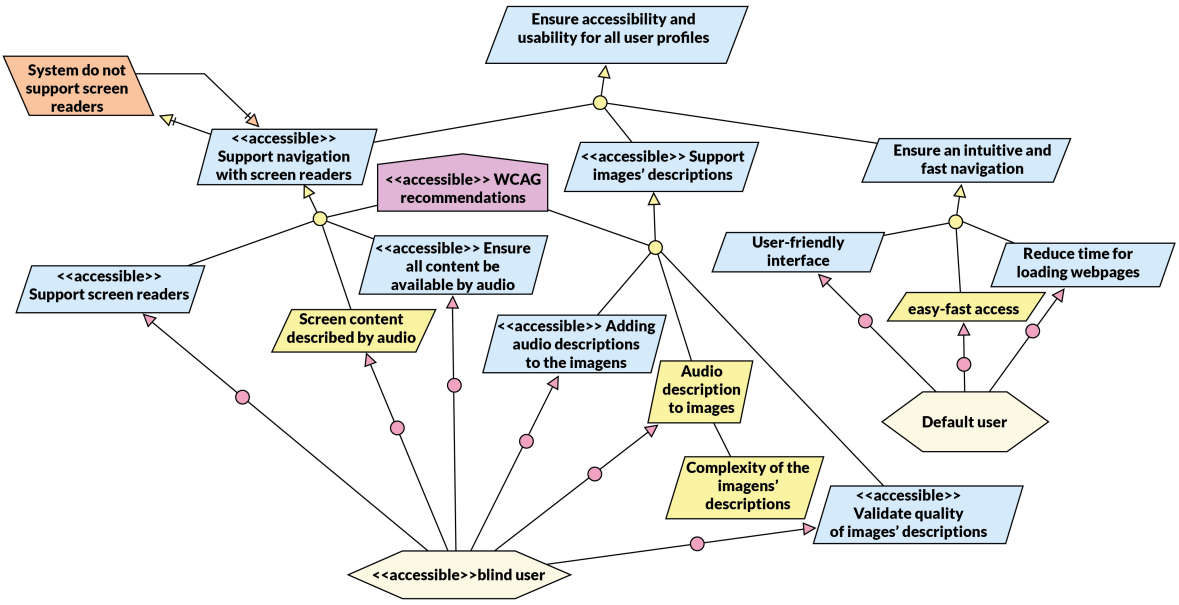

Figure 6: The goal model of the KAOS extension to represent accessibility requirements.

If feasible, the extension should be supported by a modelling tool. To support the extension, the Extender should analyse the feasibility of using existing KAOS modelling tools (such as Objectiver[7] and DSM3-KAOS[8]). Additionally, we proposed KAOS4Ext[9], a new web modelling tool with support to add the new extensions constructs' to KAOS models. This new tool enables the addition of new graphical or textual representations to create new nodes and links.

[7]https://objectiver.com/index.php?id=4
[8]https://www.cin.ufpe.br/ jhcp/dsm3goals/kaos.html
[9]https://kaosforge.com.br/

The Extender should test the modelling tool to ensure it allows and forbids the expected operations within the extension. Finally, if a suitable tool is identified, the Extender should make the tool available to other users by publishing it in a repository such as GitHub. This revised version maintains the core information and ideas while presenting them in a more continuous and readable format.

Regarding the new KAOS extension for accessibility, we included the new constructs in the KAOS metamodel by specialisation of existing Agent, Requirement, operation, and domain property constructs. The extension metamodel can be found in the complete extension specification. It was not required to propose validation rules.

The concrete syntax represented the constructs through textual stereotypes associated with their specialised constructs. The extension impacts the four (4) KAOS diagrams in which the base constructs appear. We analyzed the extension using the checklist for verification of problems and did not find any issues with completeness, consistency, or conflicts. Figure 6 presents an example of a goal model created with the proposed extension, in which we can identify accessible agents, accessible requirements, and accessible domain properties.

3.6. Validate and Evaluate the KAOS Extension (Sub-Process 4)

The sub-process presented in Figure 7 comprises three pairs of analysis and the corresponding corrections for the KAOS extension. Initially, the Extender conducts the first validation using the proposed extension to model a system. The Extender should select non-trivial and realistic examples to demonstrate the extension's capabilities. This process includes documenting the steps required to use the extension effectively. The necessary adjustments are made if the Extender identifies any corrections or improvements during this phase, such as missing relationships between concepts. In the second pair, the KAOS extension is submitted to domain experts (in KAOS extensions and the application domain) for their analysis and feedback. Given that LLMs can be utilized during the process, we emphasize the importance of evaluation by at least one KAOS expert to verify the proposed KAOS extension and identify potential errors. The third pair focuses on evaluating the effectiveness of the extension. This can be achieved through various methods, including experiments, qualitative studies, or surveys.

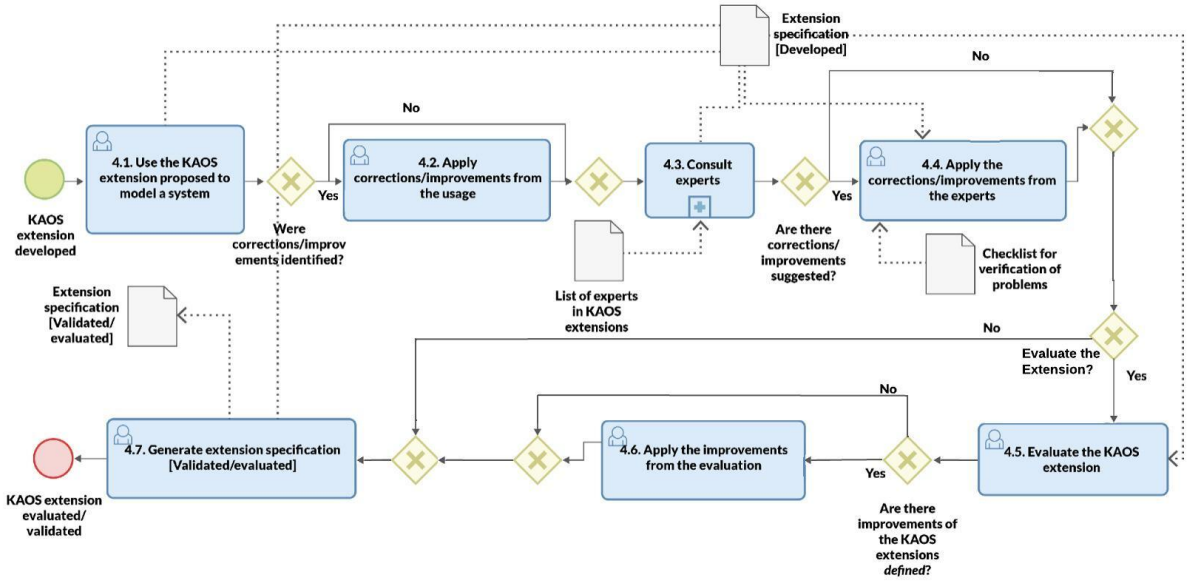

Figure 7: PRAOS Sub-process 4 Validate and Evaluate the KAOS Extensions

If the evaluation identifies any areas for improvement, the necessary corrections are applied. Artefacts used in subprocess 4:

- List of KAOS experts: List of the foremost KAOS experts to be contacted if there is a need to clarify issues.

- Extension specification (Validated/Evaluated): Generation of the extension specification.
- Problem verification checklist: Checklist with the main problems to ensure that recognised problems are not present in the extension proposal.

Regarding the extension proposed with PRAOS, we modeled the accessibility for people who are blind as an example. Figure 6 presents the goal model to our example. Additionally, a survey was conducted to evaluate the proposed extension among researchers in the area of human-computer interaction and assess their opinions on the proposed extension. The study included 20 participants from the Brazilian HCI community. The survey addressed issues related to the need to create the proposed extension, its usefulness, feasibility, impact on the HCI community, and recommendations for its use. The possible responses ranged from one (1) to five (5) on a Likert scale. Responses ranged from three (3) to five (5) for all items evaluated, demonstrating evidence of community acceptance of the proposed extension. The detailed evaluation can be found in the complete extension specification.

3.7. Check Other New Constructs to be Introduced (Task 5)

This task involves identifying the necessary constructs for the KAOS extension that were not initially identified in the first subprocess and were subsequently discovered in subprocesses 2, 3, or 4. When new constructs are identified, it becomes necessary to iterate through subprocesses 2, 3, and 4 again to incorporate them into the extension. We did not find any additional constructs during the proposed extension for accessibility.

3.8. Publicise the KAOS Extension (Sub-Process 6)

This sub-process intends to increase the visibility of new extensions within the research community. Artifacts used in sub-process 6 are the List of KAOS experts and the Extension specification (validated/evaluated). This sub-process begins with updating the KAOS extension catalog. The Extender links the related publication or updates the existing Extension Specification.

Additionally, the Extender is required to provide essential information about the extension, including title, abstract, application area, extension base, level of extension, compatibility between metamodel and concrete syntax, metamodel completeness, concept definitions, proposed construction method, reasoning approach, tool support, validation methods, validation rules, and added constructs.

If experts in KAOS extensions participated in developing the new extension, they would endorse it. Otherwise, the Extender notifies relevant experts about the new extension. The Extender should contact at least one expert in KAOS extensions to inform them about including the new extension in the catalog. These contacted experts can endorse the KAOS extension if they deem it well-defined.

In parallel, the Extender actively seeks to publish the KAOS extension. This may involve presenting the extension at conferences, publishing it in journals, or disseminating it through blogs or discussion lists. Finally, the sub-process concludes when the KAOS extension has been published and endorsed. Consequently, the extension proposal is considered complete.

An expert in KAOS extension was part of the proposal for the new KAOS extension. He endorsed the new KAOS extension. Finally, this paper is the final step in this proposal, as one of its goals is to publicize the new KAOS extension.

4. EVALUATION

We conducted a qualitative study based on semi-structured interviews to identify the perspectives about PRAOS from researchers who have proposed KAOS extensions. An interview script containing opened questions was developed and validated by a researcher experienced in goal-oriented modelling languages and extensions. We also conducted a pilot study with two undergraduate students in computing from the Federal University of Ceará, who had no prior knowledge of PRAOS or KAOS, and a pilot study with a master's student in computing who was familiar with KAOS extensions. These previous analyses

Table 1
Interview Script for the PRAOS Evaluation

1. Profile questions: How many years of experience do you have using KAOS? How many KAOS extensions have you proposed? What is your current position (Professor/developer)? Do you work in academia, industry, or both?
2. Do you know of any process for creating KAOS extensions? Which ones?
3. What is your opinion about PRAOS?
4. What are the strengths of PRAOS?
5. What are the weaknesses of PRAOS?
6. How difficult is it to understand PRAOS? What are the hardest and easiest parts of PRAOS?
7. Do you consider PRAOS necessary/useful? Why? Is PRAOS suitable/important to support the proposal of KAOS extensions by the KAOS community? Why? For whom (beginners, experts, or both)?
8. How do you think the process would be used to create extensions? As proposed or with changes/part of it?
9. Do you have any suggestions/observations related to PRAOS?
10. Is there anything about PRAOS that we did not mention in the interview that you would like to comment on?

of the script helped to improve it. The pilot data were not considered in the study. The final version of the interview script is available in Table 1. Participants watched a video about PRAOS in Portuguese, English and/or French after question 2.

The universe of this research (population) comprises authors of KAOS extensions. Thus, our universe (population) is 113 authors of KAOS extensions. We selected a sample of 13 researchers with more experience in KAOS extensions based on the analyses presented in the KAOS extensions RSL (Section 4). We obtained confirmation of participation from five (5) members of the sample, who are our participants.

The participants are geographically distributed: one (1) from Brazil, two (2) from Italy, one (1) from Portugal, and one (1) from France. The five (5) participants have a PhD, four (4) professors, and one is a technology director and professor at a company. Regarding the experience with KAOS, three participants used KAOS for 8, 9, and 15 years, respectively. The other two commented that their experience was specific to the model concepts of a particular domain. Regarding the number of extensions each participant produced, only P2 proposed two extensions, while the others produced only one (1).

The interviews were conducted via Google Meet, and each participant was interviewed individually. Each interview was recorded with the participant's consent. Each interview had an average duration of 30 minutes. The qualitative data from the interviews were then analysed using Merriam's [14] basic qualitative research procedures, in which we studied the answers to the questions and presented the main highlights.

4.1. Analysis

No participant reported knowing of any other method to extend KAOS. When asked about PRAOS, P5 commented: "PRAOS covers all the steps necessary to carry out an extension systematically and produce consistent quality results. The process is also quite complete".

Regarding the level of effort and difficulty in understanding the process, the participants found it easy to comprehend. All participants commented that they would use the method with changes. The main modifications mentioned were not consulting experts and skipping some steps, as seen in P2's comment: "I think it is possible to remove the expert from the group completely." Although some participants commented on the difficulty of getting experts in touch, one participant highlighted the value of this step.

The value observed in the use of PRAOS is not only due to the rigorous methodology, according to some participants, but also because there are certain domains in which this rigor is essential, such as in security. A general analysis of the interview's points to predominantly positive feedback, with several essential suggestions and observations that could significantly contribute to improving the process. Here are the main points highlighted:

- **Overall Approval of the Process:** The participants expressed satisfaction with the process

developed for creating new KAOS extensions. They considered the process well-structured and comprehensive, with a straightforward approach to defining and implementing new constructs.

- **Relevance of Adapted iStar Guidelines:** It has been widely acknowledged that adapting the iStar guidelines for KAOS is a valuable addition. Participants believe that these guidelines help create a solid foundation for extension development by aligning the process with established and well-known practices in goal-oriented modeling.
- **The usefulness of Guidelines and Structure:** The PRAOS structure and included guidelines were considered appropriate and helpful. Participants appreciated the organization of the process into clear sub-processes and the detailed steps that ensured the effective creation of extensions.
- **Adjustments to the Applicability of the Guidelines:** Some participants suggested adjustments to the adapted guidelines to better reflect the particularities of KAOS. Although the iStar guidelines were well adapted, there was a recommendation to further fine-tune the policies to fit the specific context and needs of KAOS.
- **Inclusion of Practical Examples:** It was suggested that including practical examples of how the new constructs were applied in real cases would clarify the practical application of the guidelines and facilitate understanding.
- **Expert Support and Collaboration:** The researchers highlighted the importance of continued collaboration with experts and suggested creating a forum or discussion group to promote the exchange of knowledge and experiences among KAOS extension developers.

4.2. Threats to Validity

In this section, we describe the threats to Criterion, Construct, Face, and Content validity, as outlined by Kitchenham and Pfleeger [12].

Criterion Validity: We identified a previous qualitative study (Gonçalves et al., 2020) with a similar objective. However, the survey by Goncalves et al. [9] aims to analyze a process proposed to support iStar extensions, whereas this one analyzes a process proposed to support KAOS extensions. Thus, the instruments used in these studies share similarities and yield satisfactory results for their respective purposes. **Construct Validity:** We recorded the audio during the interviews to enable its transcription and analysis, so we asked for permission at the beginning of the interview. This information could inhibit the participants' responses. We mitigated this threat by informing the participants that the audio files and transcripts would be kept confidential and anonymous. We also presented a confidentiality and privacy term. **Face Validity:** The interview script was tested with two undergraduate computer science students without experience with KAOS, its extensions, or PRAOS. They provided valuable feedback on the script, which led us to make corrections for better understanding. **Content Validity:** The script was tested through three pilots. We recognized this limitation and mitigated it by evaluating the script with the help of a researcher experienced in goal-oriented modelling languages and extensions.

5. CONCLUSION AND FUTURE WORK

The KAOS modelling language has been widely used and extended since its creation in the 90's. However, these extensions were developed non-systemically, resulting in problems such as specification of the extension only at the concrete syntax level, inconsistency between syntaxes, and lack of support from CASE tools. The process to support the creation of KAOS extensions was proposed based on an existing method for another goal-oriented requirements language, the PRISE process. With the adaptations ready, the new process, called PRAOS, outlines the key characteristics of creating KAOS extensions. We then proceeded to validate PRAOS. To this end, a new KAOS extension was created using PRAOS as a base. The result was an extension aimed at supporting accessibility requirements modelling. Finally, we performed a qualitative study with researchers experienced in KAOS extensions.

As future work, we propose a tool to support the use of PRAOS and another to facilitate the application and modeling of the new constructs proposed by the extension.

Acknowledgements

This work is supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT- Fundação para a Ciência e a Tecnologia, through national funds

Declaration on Generative AI

During the preparation of this work, the author(s) used Gemini and Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] M. Brambilla, J. Cabot, M. Wimmer, *Model-driven software engineering in practice*, Morgan & Claypool Publishers, 2017.

[2] R. Braun, H. Schlieter, M. Burwitz, W. Esswein, *BPMN4CP revised-extending BPMN for multi-perspective modeling of clinical pathways*, Hawaii International Conference on System Sciences, 2016.

[3] P. Caire, N. Genon, P. Heymans, D. Moody, *Visual notation design 2.0: towards user comprehensible requirements engineering notations*, 21st IEEE International Requirements Engineering Conference (RE), pp 115-124, 2013.

[4] P. Carvalho Junior, E. Gonçalves, R. Carvalho, M. A. De Oliveira, *A catalogue of KAOS extensions*, 25th Workshop on Requirements Engineering, 2022.

[5] A. Dardenne, A. van Lamsweerde, S. Fickas, *Goal-directed requirements acquisition*, Science of Computer Programming 20, 3-50, 1993.

[6] M. Fowler, *UML distilled: a brief guide to the standard object modelling language*, Addison-Wesley, 2nd Edition, 2003.

[7] R. France, B. Rumpe, *Model-driven development of complex software: a research roadmap*, Future of Software Engineering. IEEE Computer Society, 2007.

[8] E. Gonçalves, L. Monte, S. Souza, M. Oliveira, J. Araujo, *A systematic literature review of KAOS extensions*, 31st International Working Conference on Requirements Engineering Foundation for Software Quality, 2025.

[9] E. Gonçalves, J. Araujo, J. Castro, *PRISE: a process to support iStar extensions*, Journal of Systems and Software, Elsevier, v. 168, 2020.

[10] A. van Lamsweerde, *The KAOS meta-model: ten years after*, University Catholique Louvain, 1993.

[11] S. Kelly, J. Tolvanen, *Domain-specific modelling: enabling full code generation*, John Wiley & Sons, 2008.

[12] B. Kitchenham, S. Pfleeger, *Principles of survey research*, software engineering notes, v. 27, n. 5, pp. 1- 20, 2002.

[13] R. Matulevicius, P. Heymans, A. L. Opdahl, *Ontological analysis of kaos using separation of references*, Contemporary Issues in Database Design and Information Systems Development: IGI Global, 2007.

[14] S. Merriam, *Qualitative research: a guide to design and implementation*, Jossey-Bass, 2009.

[15] R. Matulevicius, P. S. Heymans, *Visually compelling goal models using KAOS*, International Conference on Conceptual Modelling, p. 265-275, 2007.

[16] S. Pfleeger, *Software engineering: theory and practice*, Pearson, 2003.

[17] L. J. R. Stroppi, O. Chiotti, P. D. Villarreal, *Extending BPMN 2.0: method and tool support*, Springer-Verlag Berlin Heidelberg, 2011.

[18] A. Sutcliffe, P. Sawyer, *Modeling personalized adaptive systems*, 25th International Conference on Advanced Information Systems Engineering, 2013.

[19] *W3C: What is Acessibility?* Available at: https://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.html, last accessed 2024/11/10.