Software and Systems Modeling
https://doi.org/10.1007/s10270-024-01254-1

THEME SECTION PAPER

Refactoring goal-oriented models: a linguistic improvement using large language models

Nouf Alturayeif[1,2] · **Jameleddine Hassine**[1,3]

Received: 24 March 2024 / Revised: 21 November 2024 / Accepted: 29 November 2024

Abstract
Goal-oriented requirements engineering (GORE) facilitates effective communication and collaboration between stakeholders. Using goal models, GORE provides a structured approach to elicit, analyze, and manage requirements from the perspective of stakeholders' goals and intentions. However, goal models are prone to several poor practices, called bad smells, which can obstruct effective communication between stakeholders. As a result, there might be misinterpretations and inconsistencies in the requirements. Goal models are particularly prone to linguistic bad smells, encompassing unclear or ambiguous goal statements, conflicting or contradictory requirements, and occurrences of misspellings. It is therefore imperative that linguistic bad smells are identified and addressed in goal models to ensure their quality and accuracy. In this paper, we build upon our previous research by enhancing the catalog of 17 goal-oriented requirements language (GRL) linguistic bad smells. We refine the detection techniques using a combination of NLP-based and LLM-based techniques. These enhancements significantly improved the tool's detection capabilities compared to our previous work. Furthermore, we offer automated refactoring solutions for 9 of these bad smells through GPT prompts. The remaining four identified bad smells are left to the user's discretion for refactoring, due to their subjective nature. The detection and refactoring processes are implemented in a tool, tailored to the Textual GRL (TGRL) language. We evaluated the bad smells refactoring approach and tool by administering a questionnaire to 13 participants, who assessed the correctness of the refactoring of 71 linguistic bad smells found in four (4) TGRL models. Participants perceived the refactored sentences as highly correct across the different types of linguistic bad smells.

Keywords Textual goal-oriented requirement language (TGRL) · Linguistic bad smells · NLP · Refactoring · GPT · LLM

1 Introduction

In virtually every software development endeavor, achieving a comprehensive understanding of stakeholders' intentions and system requirements, and articulating them accurately and unambiguously, poses a significant challenge [37]. Inadequate comprehension, analysis, and documentation of stakeholders' requirements typically result in adverse consequences downstream, jeopardizing the project's success [25]. Numerous studies have highlighted the correlation between projects exceeding their time or budget constraints and the presence of poor-quality requirements [26]. Requirements are generally specified using textual specifications and models [39]. Requirements artifacts are susceptible to poor practices, known as *bad smells*, which can compromise the quality and effectiveness of the software development process [25]. By definition, "bad smells" are not necessarily bugs or errors but rather symptoms of poor quality that may lead to maintenance difficulties, decreased readability, or increased complexity. Identifying and addressing bad smells in textual requirements and models is crucial for improving the quality, maintainability, and overall success of software systems [42, 45]. Indeed, the quality of these models has a significant

Communicated by Lola Burgueño, Davide Di Ruscio, and Dominik Bork.

✉ Jameleddine Hassine
jhassine@kfupm.edu.sa

Nouf Alturayeif
g201901790@kfupm.edu.sa

[1] Department of Information and Computer Science, KFUPM, Dhahran, Kingdom of Saudi Arabia

[2] Computing Department, Imam Abdulrahman Bin Faisal, Dammam, Kingdom of Saudi Arabia

[3] Interdisciplinary Research Center for Intelligent Secure Systems, KFUPM, Dhahran, Kingdom of Saudi Arabia

Published online: 17 January 2025

impact on the later stages of the software development process as problems in this stage disseminate to all subsequent stages in the software development process [27].

Goal-oriented requirements engineering (GORE) is dedicated to capturing and analyzing the objectives of Socio-Technical Systems (STSs), which involve intricate interactions between human actors and technical systems [21]. Its goal is to comprehend stakeholders' intentions, needs, and desires and translate them into precise system requirements. Goal-oriented requirements languages are tailored to specify and represent goals, requirements, and their relationships through goal models, offering a formal syntax and semantics for expressing and documenting desired goals and their dependencies. Various goal-modeling languages have been proposed, including *i** [52], AGORA [24], and the goal-oriented requirements language (GRL) [22] part of ITU-T's User Requirements Notation (URN) standard.

The primary objective of goal models is to facilitate transparent communication among stakeholders, developers, and project managers. However, like other requirements artifacts, goal models are susceptible to *bad smells*, or *anti-patterns*, affecting both their syntactic and semantic aspects. Identifying and rectifying these undesirable traits would improve the quality of these models. Several approaches have been developed to detect bad smells in goal models [7, 10, 13, 17, 18, 33–35, 44], mainly focusing on structural aspects and overlooking the linguistic dimension.

Given that elements in goal models are typically expressed in natural language, linguistic quality is crucial for ensuring clear and effective communication among stakeholders, fostering a shared understanding of system objectives and requirements. Despite notable advancements in Natural Language Processing (NLP) techniques, there has yet to be a comprehensive implementation of NLP methods aimed at enhancing the linguistic quality of goal models.

The i* language guide [2] provides guidelines designed to enhance the consistency and efficiency of the i* modeling process. Deviations from these guidelines are considered as indicators of bad smells. In Alturayeif and Hassine [3] and in this paper, we have integrated several of these guidelines into the development of a catalog of linguistic bad smells. We have proposed seventeen (17) linguistic bad smells in goal models that can be classified into four (4) categories: *Syntax*, *Semantics*, *Pragmatics*, and *Complexity* [3]. Each identified bad smell was exemplified with an illustrative example. We have delineated the methodology for detecting twelve (12) out of the seventeen (17) proposed bad smells. The proposed detection techniques are implemented in a tool tailored to the textual goal-oriented requirements language (TGRL) [1]. Once these bad smells have been detected, it is equally important to consider how they can be effectively addressed. This leads us to the concept of refactoring, defined as "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure" [14]. Most existing model repair approaches focus on three main types: (1) Structural repair: Addresses inconsistencies in the model's structure, such as incorrect relationships or missing elements, (2) Behavioral repair: Fixes issues related to the model's behavior, ensuring it aligns with the expected functional requirements, and (3) Semantic repair: Corrects deviations from the intended meaning or purpose of the model elements, aligning them with domain-specific knowledge [8, 30]. However, these techniques largely overlook the linguistic quality of the text within the models.

The main goal of this paper is to extend our previous work [3] by proposing and implementing refactoring techniques to address the identified bad smells. To the best of our knowledge, no comprehensive NLP-based refactoring approach has been specifically designed to improve the linguistic quality of goal models. This paper fills that gap by introducing a novel perspective on GRL models, identifying and defining various linguistic bad smells within their textual components. It presents an automated method for detecting and refactoring these issues, thereby enhancing the clarity and precision of the models. By addressing the often-overlooked linguistic quality of goal models, this work makes a significant contribution to the field and opens up new possibilities for the repair and improvement of goal-oriented models.

In this paper, we make the following contributions:

- Refine the description and the detection of the "conflicting elements" bad smell by taking into account the structure of the goal model and the interconnections among the affected elements.
- Utilize GPT to enhance the detection techniques of six (6) linguistic bad smells (incorrect actor syntax, incorrect goal syntax, incorrect softgoal syntax, incorrect task syntax, incorrect resource task, misspelled element), out of the twelve (12) techniques introduced in Alturayeif and Hassine [3]. The enhanced detection techniques achieved a precision of 0.575, a recall of 1, an F1-score of 0.74, and an F2-score of 0.875, representing a good improvement over the results obtained in Alturayeif and Hassine [3].
- Propose LLM-based refactoring techniques to address nine (9) detected linguistic bad smells (lengthy element, complex element, punctuation-market element, incorrect actor syntax, incorrect goal syntax, incorrect softgoal syntax, incorrect task syntax, incorrect resource task, misspelled element). The refactoring of similar, mismatching, and conflicting elements are delegated to the user due to their subjective nature.

- Develop a tool[1] supporting the detection and refactoring of twelve (12) linguistic bad smells in Textual GRL (TGRL) [1] models.
- Evaluate the bad smells refactoring approach and tool by administering a questionnaire to 13 participants, who assessed the correctness of the refactoring of 71 linguistic bad smells found in four (4) TGRL examples[2] from different domains.

The rest of this paper is organized as follows: Sect. 2 provides a brief introduction to goal-oriented requirements language (GRL) and its textual representation TGRL. The related work is presented in Sect. 3. In Sect. 4, we describe the linguistic bad smells present in goal models and their corresponding detection methods. The refactoring of the identified bad smells are presented in Sect. 5. Section 6 presents our TGRL-based bad smells detection and refactoring tool. Section 7 describes the empirical evaluation. Section 8 presents a discussion on the development of an integration framework to facilitate the adaptability and generalizability of our approach across different goal modeling methods. The potential threats to validity are presented in Sect. 9. Finally, conclusions and future work are presented in Sect. 10.

2 GRL in nutshell

In this section, we introduce the goal-oriented requirement language (GRL) [22] graphical syntax along with its textual representation [1] using a GRL model describing the objectives of an electric car manufacturer and an electric car owner.

2.1 GRL graphical syntax

Figure 1 illustrates the graphical GRL representation of the Electric Vehicle GRL model. Please note that some linguistic bad smells were intentionally included in the GRL model for the purpose of the empirical analysis, as discussed in Sect. 7. The model is composed of two GRL actors "Electric Car Manufacturer" and "Electric Car Owner". Actors (illustrated as) embody the intentions of stakeholders and systems. They are often used to represent stakeholders (as in this example) as well as systems.

GRL actors often enclose intentional elements that outline their objectives and capabilities. For instance, the actor "Electric Car Manufacturer" seeks to reduce production costs, increase market share, and achieve customer satisfaction. These objectives are described as three root goals (illustrated as). Softgoals differ from (hard)goals in that they lack clear-cut criteria for satisfaction. Decomposition links (depicted as , representing activities and potential solutions (or operationalizations) to achieve (hard)goals or softgoals. For instance, the goal "Expand Product Line" (in actor "Electric Car Manufacturer") is decomposed, using an AND-decomposition, into two GRL tasks "Develop New Car Models" and "Diversify Product Offerings". Actor "Electric Car Manufacturer" has 12 leaf tasks, whereas actor "Electric Car Owner" has 6 leaf tasks and one single leaf goal.

Moreover, intentional elements, within an actor, may be connected using contribution links (illustrated as), Help (), Unknown (), Break ()) or quantitative contribution weights (e.g., an integer value ranging from –100 to 100). For example, the task "Reduce Carbon Emission" (within actor "Electric Car Manufacturer") contributes positively (i.e., "Help" as qualitative value and +25 as quantitative value) to the achievement of softgoal "Minimize Environmental Impact", whereas the goal "Optimize Charging" (within actor "Electric Car Owner") contributes negatively (i.e., "Hurt" as qualitative value and –25 as quantitative value) to the achievement of goal "Extend Battery Lifespan".

The interaction between actors "Electric Car Manufacturer" and "Electric Car Owner" are expressed using three explicit dependency links (illustrated as : (1) goal "Achieve Customer Satisfaction" depends on goal "Good Vehicle Reliability", (2) goal "Achieve Customer Satisfaction" depends on resource (illustrated as) "Manage EV Fleet", which depends on softgoal "Minimize Operating Costs", and (3) goal "Good Vehicle Reliability" depends on softgoal "Enhance Vehicle Reliability".

2.2 GRL textual syntax

The introduction of TGRL [1] aims to enhance the usability, productivity, and scalability of GRL models. TGRL is delineated as an *Xtext* grammar, facilitating the generation of concrete graphical GRL syntax compatible with the *jUCMNav* tool [23]. Figures 2 and 22 illustrate the TGRL syntax corresponding to the GRL model of Fig. 1, produced using *jUCMNav* tool [23].

The TGRL specification, saved as a.xgrl file, starts with the keyword grl followed by the model's name. A TGRL specification is composed of actors. Each actor commences with the keyword actor followed by a unique actor ID. Actor names are described using the attribute name within the actor element in TGRL. Intentional elements are expressed using keywords goal, softGoal, and task. Each intentional element is characterized by a unique ID and a name. Intentional elements can be refined using a decomposition link attached to it. Such refinement is expressed using the decompositionType keyword taking "or" and "and" as values. For instance, an OR-decomposition is attached to the softgoal EnhanceBrandReputation. The sub-elements forming the refinement

[1] https://huggingface.co/spaces/nouf-sst/TGRL-bad-smells.

[2] https://github.com/Noufst/TGLR-Bad-Smells.

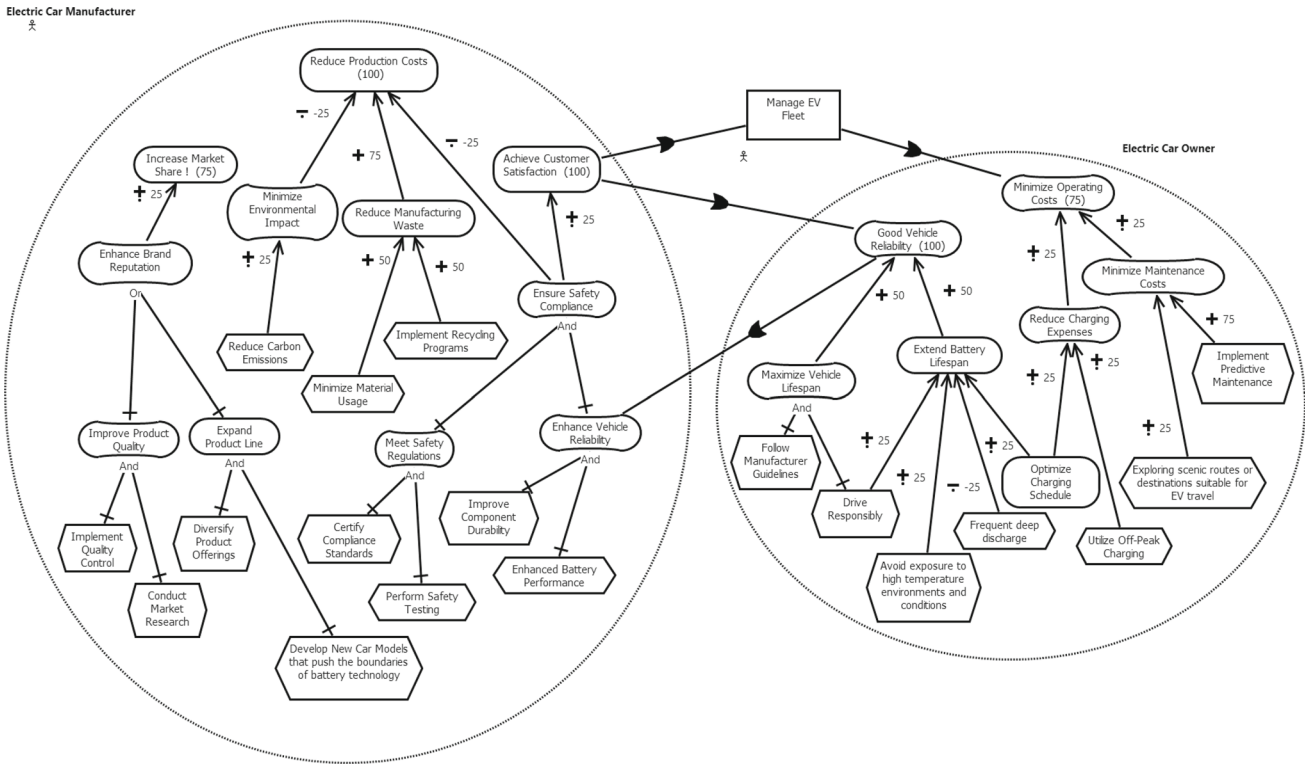


Fig. 1 Electric Vehicle GRL model

are then described using the keyword decomposedBy. For instance, the statement "EnhanceBrandReputation decomposedBy *ImproveProductQuality*, *ExpandProductLine*;", in Fig. 2, expresses such decomposition.

Contributions are denoted by the keyword contributesTo. For example, the following statement "EnhanceBrandReputation contributesTo *IncreaseMarketShare* 25;;" signifies that the softgoal EnhanceBrandReputation positively influences the goal IncreaseMarketShare with a *help*, quantified as +25.

Dependencies are expressed using the dependsOn keyword linking one source elements to an element of a different actor. For instance, the following statement "AchieveCustomerSatisfaction dependsOn *ElectricCarOwner.MinimizeOperatingCosts*;" denotes a link from the goal "AchieveCustomerSatisfaction", part of the current actor "ElectricCarManufacturer", to the softgoal "MinimizeOperatingCosts", part of actor *ElectricCarOwner*.

It's important to highlight that the resource "Manage EV Fleet" isn't bound with any actor. As a result, it's included in the GRL specification, as depicted in Fig. 22, following the definition of the "Electric Car Owner" actor.

In this research, our proposed approach is implemented specifically for TGRL [1] specifications. For a complete description of GRL and the TGRL languages, interested readers are referred to the URN standard [22] and to Abdelzad et al. [1], respectively.

3 Related work

Linguistic anti-patterns (LAs), as defined by Arnaoudova et al. [6], refer to inconsistencies in the naming, documentation, and implementation of code entities. Arnaoudova et al. [6] categorized LAs into two groups: those affecting methods and those impacting attributes. Despite their significance in code, linguistic anti-patterns have been largely overlooked in the context of models.

In this section, we review existing research pertaining to goal models bad smells detection and refactoring.

Espada et al. [13] proposed a metrics-based analysis framework supporting the quantitative assessment of complexity and completeness of KAOS goal models. The framework uses the Goal-Question-Metric (GQM) approach to derive 5 completeness metrics, namely PLGWA (Percentage of Leaf Goals With an Agent), PLGWO (Percentage of Leaf Goals With an Object), PLOWS (Percentage of Leaf Obstacles With a reSolution), PLGWOp (Percentage of Leaf Goals With an Operation), POpWA (Percentage of Operations With an Agent), and 4 complexity metrics, namely ANLG (Number of Leaf Goals per Agent), GNO (Number of Objects per Goal), MD (Model Depth), and RNSG (Root Number of Sub-Goals). The metrics are formalized using OCL and implemented in *modularKAOS*, an existing tool for editing KAOS goal models. The metrics suite was evaluated with real-world case studies.

Fig. 2 TGRL specification of the electric vehicle model (Electric Car Manufacturer)

```
grl ElectricVehicle {
  actor ElectricCarManufacturer{
      name = "Electric Car Manufacturer";
      // ------- SoftGoals -------
      softGoal EnhanceBrandReputation {
          name = "Enhance Brand Reputation";
          importance = high;
          decompositionType = or;
      }
      softGoal ImproveProductQuality {
          name = "Improve Product Quality";
          decompositionType = and;
      }
      softGoal MinimizeEnvironmentalImpact {
          name = "Minimize Environmental Impact";
      }
      softGoal EnhanceVehicleReliability {
          name = "Enhance Vehicle Reliability";
          decompositionType = and;
      }
      softGoal MeetSafetyRegulations {
          name = "Meet Safety Regulations";
          decompositionType = and;
      }
      softGoal EnsureSafetyCompliance {
          name = "Ensure Safety Compliance";
          decompositionType = and;
      }

      // ------- Goals -------
      goal IncreaseMarketShare {
          name = "Increase Market Share !";
      }
      goal ReduceProductionCosts {
          name = "Reduce Production Costs";
      }
      goal ExpandProductLine {
          name = "Expand Product Line";
          decompositionType = and;
      }
      goal ReduceManufacturingWaste {
          name = "Reduce Manufacturing Waste";
      }
      goal AchieveCustomerSatisfaction {
          name = "Achieve Customer Satisfaction";
          importance = 100;
      }

      // ------- Tasks -------
      task ImplementQualityControlMeasures {
          name = "Implement Quality Control Measures";
      }
      task ConductMarketResearch {
          name = "Conduct Market Research";
      }
      task DevelopNewCarModels {
          name = "Develop New Car Models that push the boundaries of battery technology";
      }
      task DiversifyProductOfferings {
          name = "Diversify Product Offerings";
      }
      task ReduceCarbonEmissions {
          name = "Reduce Carbon Emissions";
      }
      task MinimizeMaterialUsage {
          name = "Minimize Material Usage";
      }
      task ImplementRecyclingPrograms {
          name = "Implement Recicling Programs";
      }
      task CertifyComplianceStandards {
          name = "Certify Compliance Standards";
      }
      task PerformSafetyTesting {
          name = "Perform Safety Testing";
      }
      task ImproveComponentDurability {
          name = "Improve Component Durability";
      }
      task EnhanceBatteryPerformance {
          name = "Enhanced Battery Performance";
      }

      // ------ Decompositions -------
      EnhanceBrandReputation decomposedBy ImproveProductQuality, ExpandProductLine;
      ImproveProductQuality  decomposedBy ImplementQualityControlMeasures, ConductMarketResearch;
      ExpandProductLine decomposedBy DevelopNewCarModels, DiversifyProductOfferings;
      EnsureSafetyCompliance decomposedBy MeetSafetyRegulations, EnhanceVehicleReliability;
      MeetSafetyRegulations decomposedBy CertifyComplianceStandards, PerformSafetyTesting;
      EnhanceVehicleReliability decomposedBy ImproveComponentDurability, EnhanceBatteryPerformance;

      // ------ Contributions -------
      EnhanceBrandReputation contributesTo IncreaseMarketShare {25;};
      ReduceCarbonEmissions contributesTo MinimizeEnvironmentalImpact {25;};
      MinimizeEnvironmentalImpact contributesTo ReduceProductionCosts {-25;};
      ReduceManufacturingWaste contributesTo ReduceProductionCosts {75;};
      EnsureSafetyCompliance contributesTo ReduceProductionCosts {-25;};
      MinimizeMaterialUsage contributesTo ReduceManufacturingWaste {-50;};
      ImplementRecyclingPrograms contributesTo ReduceManufacturingWaste {-50;};
      EnsureSafetyCompliance contributesTo AchieveCustomerSatisfaction {25;};

      // ------ Dependencies -------
      AchieveCustomerSatisfaction dependsOn ManageEVFleet;
      ManageEVFleet dependsOn ElectricCarOwner.MinimizeOperatingCosts;
      AchieveCustomerSatisfaction dependsOn ElectricCarOwner.GoodVehicleReliability;
  }
```

In a related work to Espada et al. [13], Gralha et al. [17, 18] have introduced, implemented, and validated metrics concerning both complexity and completeness for i* models [52]. Concerning complexity, the researchers pinpointed opportunities for refactoring aimed at enhancing the modularity of i* models, thereby mitigating their complexity. As for completeness, they devised methods to automatically identify model incompleteness, offering requirements engineers insights into the progress of their model completion.

Santos et al. [44] assessed how the layout guidelines (introduced by the i* community) influence the ability of novice i* stakeholders to comprehend and review i* models. The authors conducted a quasi-experiment in which participants were tasked with two comprehension and two review exercises. Each set of tasks involved a model with poor layout and another adhering to the i* layout guidelines. The authors gauged the impact of layouts by considering the success rates and effort required for task completion. Effort was evaluated using time, perceived complexity (using NASA TLX), and eye-tracking data. The experiment results showed that adherence to existing i* layout guidelines did not significantly influence the understanding and review performance of i* models.

GRL models, as specified by ITU-T's User Requirements Notation (URN) [22], inherently allow for the existence of *cycles*, also known as *circular dependencies*. While these cycles do not render the model invalid when used as a means of communication to document stakeholders' intentions or convey requirements, they pose challenges during satisfaction analysis due to disruptions in propagation algorithms [4, 22]. As a result, evaluating available strategies, which is integral to trade-off analysis, becomes unpredictable. Mohammed et al. [33] formalized the circular dependency bad smell in GRL models and developed an approach based on the search-based simulated annealing (SA) algorithm to detect its instances. The authors have introduced two strategies, referred to as pruning and pairing, to enhance the efficiency of their proposed method. They conducted empirical assessments on three algorithm combinations: (1) the basic simulated annealing (SA) search algorithm, (2) the SA search algorithm enhanced with pruning, and (3) the SA search algorithm augmented with both pruning and pairing mechanisms, across multiple case studies. Findings indicate that simulated annealing, when complemented with pruning and pairing, emerges as the most efficient approach in terms of computational speed and accuracy. To resolve the circular dependency issue, the analyst can break the cycle by removing one of its links. However, this refactoring task was deferred for future consideration.

Furthermore, Mohammed et al. [35] introduced and formalized four structural GRL bad smells: (1) Overly Ambitious Actor: an actor with an excessive number of objectives, (2) Overly Operationalized Actor: an actor with an abundance of tasks and resources in comparison with the low-level goals and softgoals, (3) Deep Hierarchical Refinement: an actor with a deep refinement hierarchy, and (4) Highly Coupled Element: an element with a significant number of incoming or outgoing links. The detection and identification of instances of these bad smells are facilitated by a set of metric-based rules and have been implemented as an Eclipse plugin tool, named *GSDetector*, seamlessly integrated with jUCMNav [23], recognized as the most comprehensive GRL tool currently available. More recently, Mohammed et al. [34] introduced another five structural GRL bad smells: (1) Negligible sub-model occurs when a root goal/softgoal has an importance value of zero, (2) Non-strategic sub-model is a sub-model that starts the refinement tree with a task or resource, (3) Un-Operationalized Objective occurs when the model contains a leaf goal/softgoal, i.e., goal/softgoal is not operationalized into tasks/resources, (4) Disturbed operationalization occurs when a task or a resource has a goal or a softgoal as a child, (5) Unknown contribution occurs when a contribution link has unknown or zero contribution value. The presence of these bad smells indicates violations of the constraints defined for each bad smell, which were intended to be maintained by the GRL input model. The automated detection of these bad smells is implemented within jUCMNav [23].

Moreover, the jUCMNav [23] tool encompasses a comprehensive set of guidelines tailored for the development of GRL models. These guidelines are organized into distinct categories aimed at ensuring model completeness and consistency across various dimensions. The first category prioritizes the integrity and coherence of GRL models, offering a series of rules that assess these aspects comprehensively. For instance, one guideline mandates that each actor must incorporate at least one element with a nonzero importance value. In other words, if every element within an actor possesses an importance value of zero, then this actor is in violation of the guideline. The second category pertains to the enforcement of formal constraints governing the utilization of GRL and its extensions to align it with an i* style [5]. For example, i* emphasizes full dependency relations through the inclusion of depedums. While GRL allows for diverse configurations of dependency relations, i* constrains dependency relations to full dependencies only when the dependum exists. According to the URN ITU-T standard [22], having a dependum is discretionary. However, it also suggests that including a dependum results in a more comprehensive dependency relationship. The third category addresses unused constructs, specifically targeting the undesirability of such elements. Deviation from these guidelines may be viewed as indicative of bad smells. For instance, within this category, one guideline stipulates that the GRL model should avoid including empty actors. Such instances are deemed undesirable and should thus be eliminated. Furthermore,

additional limitations stem from the inherent characteristics of the GRL language [38]. These deficiencies are unrelated to modeling practices or methodologies and may pose challenges in certain contexts.

Cares and Franch [10] proposed a semiotic-based quality assessment proposal built upon the i* framework and the SEQUAL (SEmiotic QUALity) proposal [29]. Semiotic concepts encompasses not only the documentation or models themselves, but also involves actors acting as interpreters, the presence of informality and diversity in languages, and the social dynamics involved in reaching agreements. The structure of SEQUAL emphasized the nuances of the requirements stage, while maintaining its primary focus on three key types of qualities: syntax quality, semantic quality, and pragmatic quality. The authors proposed a simplification of SEQUAL which can be applied to i* models by defining semantic, pragmatic and social metrics. The suite of metrics are implemented using *iStarML*, a XML representation of i* models. Our proposed approach for detecting and refactoring bad smells does not incorporate the analysis of the application domain or the engagement of real actors.

Asano et al. [7] investigated the quality of goal refinements in AGORA goal models [24]. Four undesirable bad smells, named "symptoms" in their study, were identified, along with proposed detection methods. The first symptom, labeled as "low semantic relations between a parent and its children" means the lack of relevance between a child and its parent. Detection of this symptom involved utilizing a lightweight NLP technique, known as *case frames*, where goal descriptions are depicted as case frames comprising verbs and co-occurring words, referred to as concepts. These concepts are then employed to ascertain the various meanings of a verb, subsequently mapped into a hierarchical dictionary to compute similarities between a goal and its children. Detection of low semantic relations is achieved by comparing these similarities against a predefined threshold. The second and third undesirable symptoms, "too many siblings" and "too few siblings" pertain to the adequacy of the number of children in achieving the parent goal. Detection of these symptoms involves utilizing a metric based on the number of children, alongside upper and lower thresholds. The upper threshold denotes the maximum permissible number of children associated with a parent; exceeding this threshold constitutes an instance of the "too many siblings" trait. Conversely, the lower threshold signifies the minimum number of children required for a parent; falling below this threshold constitutes an instance of the "too few siblings" trait. The fourth undesirable symptom, termed "course-grained leaf goal" pertains to leaf goals lacking specificity. It assesses the degree of concreteness of each leaf goal by examining the depth of refinement of each model branch. If a branch's depth is deemed insufficient, the associated leaf goal is flagged as an instance of this trait. A notable limitation of this study is the absence of clearly defined thresholds for detecting these symptoms, leaving developers to specify them independently.

Most of the surveyed approaches focus mainly on the structural aspect of goal models. The study by Asano et al. [7] is the only work that deploys an NLP (Natural Language Processing) technique. Furthermore, all surveyed techniques describe methods for detecting bad smells in goal models without offering or implementing specific techniques for refactoring.

This paper presents a new perspective by identifying and defining various linguistic bad smells and offering an automated approach for their detection and refactoring. By emphasizing the often-overlooked linguistic quality of text in models, it makes a novel contribution to the field and opens new avenues for applications in model repair.

4 Detecting linguistic bad smells in GRL models

This section describes the bad smells initially introduced in Alturayeif and Hassine [3] and their enhanced detection techniques. A total of 17 linguistic bad smells were identified [3] by leveraging insights from pertinent literature. These bad smells encompass various aspects and many of them are rooted in the best practices outlined in the i* language guide [2] and the GRL standard [22]. It is worth noting that GRL [22] is based on i* and shares many of its concepts [5].

Each bad smell is documented using: (1) a concise description, (2) an illustrative example, and (3) the relevant detection method(s) (if applicable). Automated detection methods were provided for 12 of the identified bad smells (labeled 1 to 12), leaving the remaining 5 bad smells (labeled 13 to 17) for future investigation.

4.1 Lengthy element

- *Description:* Seki et al. [45] identified "Long Sentence" as a bad smell in use case descriptions, where excessive details make sentences harder to understand and confuse readers. Detection relies on comparing sentence lengths to a threshold based on their distribution [45]. Similarly, Joseph Frantiska [15] recommended that use case names in UML diagrams be concise, ideally two or three words, to ensure clarity and precision. In our context, this bad smell denotes a wordy intentional element. A short intentional element conveys the essential intention without unnecessary details that could distract from the core objective. It was named "Element size" in Alturayeif and Hassine [3].
- *Illustrative example:* An example of a lengthy element is "Avoid Exposure to High Temperature Environments

and Conditions". It may be expressed more concisely as "Avoid High Temperatures".
– *Detection method:* The lengthy element bad smell can be detected by word count using a regular expression with a threshold.

The refactoring of this bad smell is intrinsically linked to its detection. During the detection phase, we defined a threshold to facilitate the identification of the smell, and this same threshold was subsequently used in the refactoring process. The rationale for implementing a threshold in goal models is to ensure consistency in the level of detail used to describe elements, avoiding situations where some elements are described with only 2-3 words while others are described with 20 or more. Goal models are intended to focus on high-level objectives and stakeholder intentions, rather than delving into complex requirements or detailed specifications. If an element becomes overly lengthy, it may indicate that the element is too complex and should be split into smaller, cohesive components. However, the requirements engineer can adjust the threshold by increasing it if they find that the meaning of an element is too shallow and requires further clarification. Moreover, the design prompt is intended to summarize the content of a lengthy element without altering its original phrasing.

4.2 Complex element

– *Description:* Complex text increases cognitive load, hindering information processing, especially in visual notations like goal models, where human perceptual and cognitive capacities are limited [36]. A complex sentence is a sentence with one independent clause and at least one dependent clause. In goal models, intentional elements should not be complex because simplicity enhances clarity, facilitates decomposition, and supports easier analysis and communication. It was named "Complex sentence" in Alturayeif and Hassine [3].
– *Illustrative example:* An example of a complex sentence is "develop new car models that push the boundaries of battery technology". Instead, it may be expressed as "Create innovative car models with advanced batteries".
– *Detection method:* Find the sentences constituents that has the label "SBAR", which stands for Subordinate Clause [47]. A subordinate clause, is a dependent clause that cannot stand alone as a sentence. We used the Stanza library [40] to extract the constituents' labels. Using the Stanza library [40], we perform constituency parsing to analyze the syntactic structure of each sentence, generating a hierarchical tree that captures its grammatical composition. This technique breaks down sentences into constituent parts, such as noun phrases, verb phrases, and clauses, organizing them into a tree format that reflects their syntactic relationships. Stanza's dependency parser utilizes a Bi-LSTM-based deep biaffine neural model to assign standard syntactic labels (e.g., "NP" for noun phrase, "VP" for verb phrase, and "SBAR" for subordinate clause). By traversing the parse tree, we specifically identify and extract subordinate clauses, marked by the label "SBAR".

4.3 Punctuation-marked element

– *Description:* This bad smell denotes an element with punctuation marks, such as: period, question mark, exclamation point, comma, semicolon, colon, parentheses, brackets, braces, quotation marks, and ellipsis. It was named "Punctuation Marks" in Alturayeif and Hassine [3].
Punctuation marks serve to structure writing ensuring the reader can understand the meaning and tone accurately. In the context of goal models, punctuation is often unnecessary. This is because the concise nature of the intentional elements allows the intended meaning to be easily understood without additional punctuation. For example, a task named "Increase sales" or a goal named "Cost reduction" are clear and direct, rendering punctuation redundant. Furthermore, adding punctuation can introduce unnecessary confusion (with respect to the priority or the uncertainty of an element), detracting from the directness and simplicity that short text aims to achieve. For example, in the Non-functional Requirements (NFR) framework [12], priority softgoals are marked with an exclamation point (!), while the Collaborative Systems Requirements Modeling Language (CRML) [48] uses one to three exclamation marks to denote task importance. However, relying on punctuation for prioritization is ineffective and prone to misinterpretation. Instead, more robust approaches like priority attributes (e.g., high, medium, low) and structured frameworks, such as MoSCoW [28], provide greater clarity and consistency.
To emphasize this aspect to the user, the tool displays the following warning message whenever this bad smell is detected: "Warning: Avoid using punctuation to imply priority or urgency. The use of punctuation can lead to misinterpretation and inefficiencies in communicating the goals/requirements".
– *Illustrative Example:* An example of a punctuation-marked element is "Increase Market Share!". Instead, it should be expressed as "Increase Market Share".
– *Detection method* The detection of the Punctuation-marked element can be achieved using a regular expression.

4.4 Incorrect actor syntax

– *Description:* An actor denotes a role, an agent (hardware or software), or a position. It cannot be expressed as a verb phrase (VP), as stipulated by the i* guide [2]: "Do not use Verb Phrases to name Actors, Agents or Positions". [3]

– *Illustrative Example:* An example of an incorrect actor syntax is "Own electric car". Instead, it should be 'Electric car owner'.

– *Detection method:* In Alturayeif and Hassine [3], this bad smell is detected using POS tagging. We have utilized a BERT model fine-tuned on POS tagging task [43]. This model was trained on the Penn TreeBank dataset [31] and achieved an F1-score of 96.69. BERT model which is more accurate than the standard POS taggers as it assigns a POS tag to each token (i.e., subword) rather than each word. To detect an incorrect actor syntax, the sentence should not start with a word with any verb POS tag. According to Penn TreeBank dataset [31], the verb POS tags are VB (base form), VBD (past tense), VBG (gerund or present participle), VBN (past participle), VBP (non-3rd person singular present), and VBZ (3rd person singular present). The performance of this method, as reported in Alturayeif and Hassine [3], meets an acceptable standard. Nevertheless, our aim is to enhance it further.

Since detecting noun and verb phrases in natural language is a complex task due to its inherent ambiguity. Large Language models (LLMs), such as GPT, have demonstrated exceptional performance across a wide range of NLP tasks due to their ability to capture and understand contextual, syntactic, lexical, semantic, and idiomatic aspects of a language. Hence, in this paper, we tried GPT to automatically detect noun and verb phrases by curating the prompt shown in Fig. 3. The prompt returns an answer indicating whether the sentence is a noun or verb phrase. Based on that, incorrect actor syntax can be reported. After multiple experimental prompts, we observe that when the GPT is asked to reason about the answer, the results are more accurate. More details on prompt-engineering of GPT are discussed in Sect. 5. The performance of the GPT detection method outperforms the BERT model fine-tuned on POS tagging.

4.5 Incorrect goal syntax

– *Description:* A (hard) goal denotes an actor's objective. A goal shall be expressed as a Noun Phrase (NP). The i* guide [2] stipulates: "Goals should not be confused with the Tasks, which start with verbs". [4]

– *Illustrative Examples:* An example of an incorrect goal syntax is "Optimize charging schedule". Instead, it may be expressed as "Charging schedule optimization" or "Optimization of charging schedule". Figure 4 illustrates the difference between a goal (expressed as a Noun phrase) and a task (expressed as a Verb phrase). The goal should be expressed as "Patients be cured" (or "Patients are cured") instead of "Cure patients", which expresses a task rather than a goal.

It is important to emphasize that the specific style of goal writing demonstrated in the examples is not universal. Different contexts and domains may employ diverse styles and conventions for articulating goals, provided they do not exhibit the symptoms of this particular bad smell.

– *Detection method:* The detection method of this bad smell is similar to the detection method of Incorrect Actor Syntax.

```
prompt = [
  {
    "role": "system",
    "content": '''
        You are a specialist in English linguistics.
        You will be provided with a sentence, and your
        task is to determine whether the sentence is a
        noun phrase or a verb phrase.
        Answer with "noun phrase" or "verb phrase" and
        your reasons.
        Use JSON format with keys "answer" and "reason".'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
  }
]
```

Fig. 3 Prompt for incorrect actor syntax detection

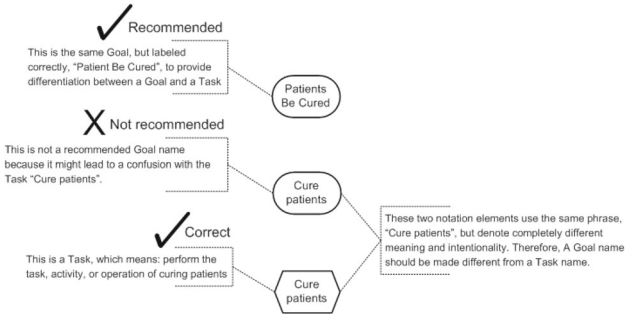


Fig. 4 Goals should not to be confused with the Tasks, which start with verbs [2]

[3] http://istar.rwth-aachen.de/tiki-browse_categories.php?parentId=20.

[4] http://istar.rwth-aachen.de/tiki-index.php?page=Goals+%28Hard+Goals%29&structure=i%2A+Guide.

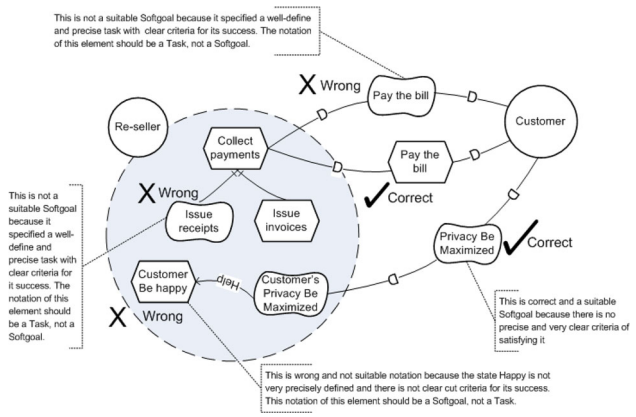


Fig. 5 Do not confuse between a Softgoal and a Task [2]

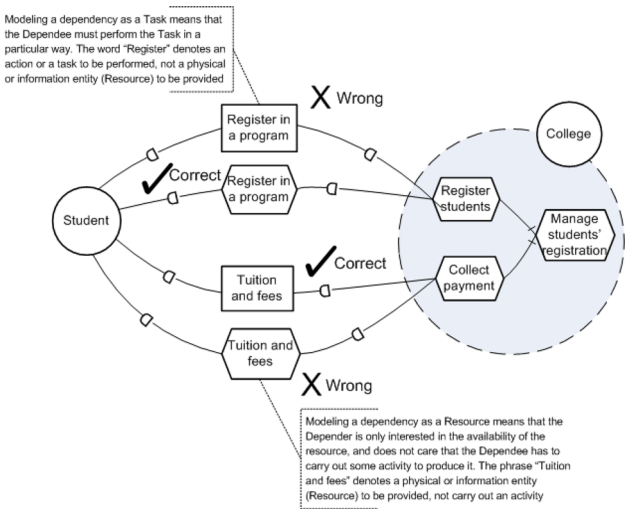


Fig. 6 Do not confuse between a Task and a Resource [2]

4.6 Incorrect softgoal syntax

- *Description:* A softgoal denotes an objective with no clear-cut criteria with respect to its satisfaction. A softgoal shall be expressed as a Noun Phrase (NP), similar to a (hard) goal, as outlined in the i* guide [2].
- *Illustrative examples:* An example of an incorrect softgoal syntax is "Minimize operating costs". Instead, it may be expressed as "Operating Costs are Minimal" or "Operating Costs are Minimized". Figure 5 illustrates the difference between a softgoal (expressed as a Noun phrase) and a task (expressed as a Verb phrase).
- *Detection method:* The detection method of this bad smell is similar to the detection method of Incorrect Actor Syntax.

4.7 Incorrect task syntax

- *Description:* A Task denotes an activity. A task shall be expressed as a Verb Phrase (VP), as stipulated by the i* guide [2] (see Sects. 4.5 and 4.6).
- *Illustrative examples:* An example of an incorrect task syntax is "Enhanced battery performance". Instead, it may be "Enhance battery performance". Figures 4 and 5 illustrate a couple of additional examples.
- *Detection method:* The detection method of this bad smell is similar to the detection method of Incorrect Actor Syntax. However, instead of reporting verb phrases as bad smells, we report noun phrases.

4.8 Incorrect resource syntax

- *Description:* A resource is defined as "a physical or informational entity, for which the main concern is whether it is available" [22]. A resource shall be expressed as a Noun Phrase (NP), as stipulated by the i* guide [2].
- *Illustrative examples:* An example of an incorrect resource syntax "Manage EV fleets". Instead, it may be "EV Fleet Management System". Figure 6 illustrates the difference between a resource and a task.
- *Detection method:* The detection method of this bad smell is similar to the detection method of Incorrect Actor Syntax.

4.9 Misspelled element

- *Description:* Misspellings can be categorized into two types: competence errors, arising from a lack of knowledge about the language, and performance errors, caused by external factors such as inattention, stress, or fatigue. These errors can result in misunderstandings, misinterpretations, and ambiguity [16]. This bad smell indicates misspelled words within a GRL element. It was named "Misspellings" in Alturayeif and Hassine [3]. It is important to note that, to our knowledge, there are no existing GRL/TGRL tools that offer spelling checks within GRL/TGRL specifications.
- *Illustrative example:* An example of a misspelled element is "Implement recicling programs". It should be "Implement recycling programs".
- *Detection method:* In Alturayeif and Hassine [3], we developed a detection method of this bad smell as follows: Given the original word w, we find the correct word c out of all possible candidate words that maximizes the probability that c is the intended word. For

```
prompt = [
  {
    "role": "system",
    "content": '''
        You are a specialist in English linguistics.
        You will be provided with a sentence and your
        task is to report any misspelled words and
        correct the spelling if needed.
        Answer with "correct" or "misspelled". In case
        the sentence is misspelled, correct it with the
        right spelling.
        Use a JSON format with keys 'original sentence',
        'answer', and 'correct sentence'.'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
   }
]
```

Fig. 7 Prompt for misspelled element

that, we used Autocorrect library[5] that is based on Peter Norvig's approach.[6] It consists of four main parts:

- *Selection Mechanism*: select the candidate with the highest probability.
- *Candidate Model*: determine the candidate words c to consider.
- *Language Model*: determine the probability that c appears as a word of English text.
- *Error Model*: denotes the probability of typing w instead of c.

However, we observed that in some cases this approach do not take into account the word acronym. For example, *"AI"* in the phrase *"AI System"* was detected as a misspelled *"As"*. In some other cases, this approach was not able to capture new words. For example, the word *"Gamify"* in the sentence *"Gamify Learning Activities"* was detected as a misspelled *"Family"*. In such cases, GPT have shown powerful capabilities as it is trained on massive amounts of text, including recent data. For that, we designed a GPT prompt to detect and correct the misspelled elements, as shown in Fig. 7.

4.10 Similar elements

- *Description:* This bad smells targets semantically duplicate/similar text used in different elements within the same actor. This would create confusion and ambiguity, undermining the clarity and distinctiveness of the model's goals and tasks.
- *Illustrative example:* An example of this bad smell would be "Good Vehicle Reliability" and "Enhance Vehicle Reliability". They are semantically and syntactically similar.
- *Detection method:* This bad smell is detected using a pre-trained deep learning model with a threshold. We utilized a BERT model fine-tuned on sentence similarity task[7] that predicts a score between 0 and 1 presenting the semantic similarity of two sentences. This model was trained on the STS Benchmark dataset [32].

4.11 Goal/task and sub-goal/sub-task mismatch

- *Description:* A sub-goal/sub-task does not follow its primary goal/task. The instance of this bad smell occurs when there is an illogical or misleading link between elements, such as a task or sub-goal that does not appropriately or realistically support the achievement of a higher-level goal, leading to confusion and misalignment within the model.
- *Illustrative example:* An example of this bad smell is a Task "Exploring scenic routes or destinations suitable for EV travel" contributing positively to "Minimize Maintenance Costs", as achieving the former element is not part of achieving the latter element.
- *Detection method:* To detect this bad smell, we use a BERT model fine-tuned on Natural Language Inference (NLI) task. NLI determines whether a *hypothesis* is true (entailment), false (contradiction), or undetermined (neutral) given a *premise*. We fine-tuned the general BERT model on NLI task using Multi-Genre Natural Language Inference (MultiNLI) corpus [49] that consists of 433k hypothesis/premise pairs. However, due to hardware limitations, we used 50k pairs. The input of the model is a goal/task (hypothesis) and a sub-goal/sub-task (premise), and the pair is mismatching if the hypothesis is false or undetermined.

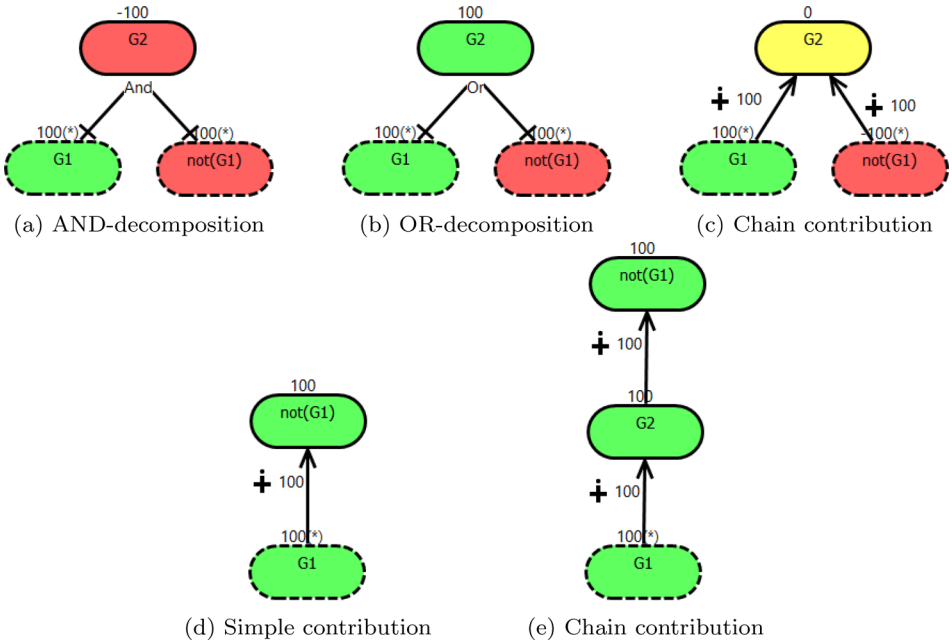
4.12 Conflicting elements

- *Description:* Two elements within an actor are conflicting with each other, which may create inconsistencies and hinder goal achievement by presenting contradictory objectives or requirements that cannot be simultaneously fulfilled. This bad smell was named "Conflicting goals/tasks" in Alturayeif and Hassine [3]. This definition is not accurate. Hence, we refine it as follows, "An actor contains two conflicting elements with no negative direct/indirect linkage between them". Figure 8 illustrates examples of smelly configurations. In all examples,

[5] https://github.com/filyp/autocorrect.

[6] https://norvig.com/spell-correct.html.

[7] https://huggingface.co/WillHeld/roberta-base-stsb.



(a) AND-decomposition (b) OR-decomposition (c) Chain contribution

(d) Simple contribution (e) Chain contribution

Fig. 8 Examples of smelly configurations involving conflicting elements (G1 and not(G1))

we denote by *G1* and *not(G1)*, the two conflicting goals. For example, Fig. 8a illustrates an AND-decomposition where the satisfaction of G1 (in green color) means necessary the denial of not(G1), leading always to the denial of G2 (in red color). Similarly, Fig. 8b illustrates an OR-decomposition where the satisfaction of G1 (in green color) means necessary the denial of not(G1), leading always to the satisfaction of G2 (in green color). Figure 8c illustrates two contributions, where the satisfaction of G1 means necessary the denial of not(G1), leading always to an unknown satisfaction of G2 (in yellow color). Figure 8d shows a simple positive contribution between G1 and not(G1). When G1 is satisfied, not(G1) is also satisfied, which is contradictory. Likewise, Fig 8e depicts a chain of positive contributions. Here, the fulfillment of G1 results in the fulfillment of not(G1), presenting a contradiction.

In presence of a negative contribution, as shown in Fig. 9a and 9b, the satisfaction of G1 would lead to the denial of not(G1), which is acceptable.

Please be aware that encountering an even number of negative contributions within a chain of contributions from two conflicting elements is improbable. However, should such a scenario occur, we recommend flagging it as a bad smell for the analyst's consideration.

– *Illustrative Example:* An instance illustrating this bad smell occurs when conflicting elements such as 'Reduce cost' and 'Upgrade hardware specifications' are linked using a positive contribution link.
– *Detection method:* In Alturayeif and Hassine [3], we employed an approach akin to the Goal/Task and Sub-goal/Sub-task Mismatch method, differing only in considering the pair as conflicting if the hypothesis is false. This assumption is flawed, as it may be acceptable for conflicting elements to coexist within the same actor.

In this paper, we refine the detection by designing an algorithm to check for linguistic conflicts only in cases where they are not allowed. These cases include: (1) *intra-conflict*: the goals/tasks are in and/or decomposition or their contributions are of the same sign, and (2) *inter-conflict*: goals/tasks are in transitive relationship and their contributions are of the same sign. Algorithm 1 presents the main steps, whereas Algorithms 2 and 3 demonstrate intra-conflicts and inter-conflicts procedures, respectively.

Algorithm 1 Check for conflicting goals/tasks

```
for all actors do
    for all pairs of goals/tasks within an actor do
        INTRA_LINGUISTIC_CONFLICTS(pair)
        INTER_LINGUISTIC_CONFLICTS(pair)
    end for
end for
```

4.13 Incomplete text

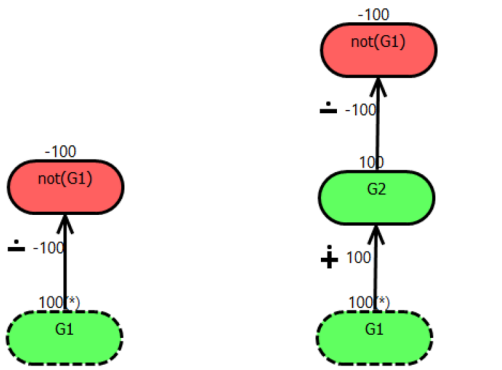
– *Description:* A text that requires additional text to be meaningful. Incomplete text can obscure the intent and meaning of goals or tasks, leading to misunderstandings and ineffective goal analysis or implementation.
– *Illustrative Example:* An instance illustrating this bad smell occurs when an intentional element has just one

Algorithm 2 Check for intra-linguistic conflicts

```
procedure INTRA_LINGUISTIC_CONFLICTS(pair)
    if the pair have at least one contribution to any other goal/task
then
        if the contributions are of opposite signs then
            continue
        else
            CHECK_LINGUISTIC_CONFLICTS(pair)
        end if
    end if
end procedure
```

Algorithm 3 Check for inter-linguistic conflicts

```
procedure INTER_LINGUISTIC_CONFLICTS(pair)
    paths = GET_ALL_PATHS_BETWEEN_PAIR(pair)
    nodes, edges = CONSTRACT_GRAPH(pair, paths)
                ▷ get the contribution values from the first and last edges
    contributions = [edges[0], edges[−1]]
    if the contributions are of opposite signs then
        continue
    else
        CHECK_LINGUISTIC_CONFLICTS(pair)
    end if
end procedure
```



(a) Simple contribution (b) Chain contribution

Fig. 9 Examples of allowed configurations involving conflicting elements (G1 and not(G1))

word, in absence of qualifier, etc. For example, "charging" is an example of incomplete text. It should be "Fast charging" or "Optimize charging".
– *Detection method:* Unavailable, as it depends on domain expertise and the goal model context.

4.14 Incorrect goal semantics

– *Description:* A (hard)goal denotes an objective with a clear-cut criteria with respect to its satisfaction. An instance of this bad smell occurs when a goal represents an objective with subjective or vague criteria for its satisfaction, allowing for a range of interpretations and fulfillment.
– *Illustrative example:* "Good vehicle reliability" isn't a defined goal due to the lack of clear criteria to determine its achievement.
– *Detection method:* Unavailable, as it depends on domain expertise and the goal model context.

4.15 Incorrect softgoal semantics

– *Description:* The URN standard [22] defines a softgoal as follows: "A Softgoal is a condition or state of affairs in the world that the actor would like to achieve, but unlike in the concept of (hard) goal, there are no clear-cut criteria for whether the condition is achieved". Generally, a softgoal is used to denote quality attributes or non-functional requirements (NFRs). An instance of this bad smell occurs when a softgoal is defined with clear-cut criteria for satisfaction, which is inconsistent with its definition.
– *Illustrative example:* "Reduce production costs" is not a softgoal as we can clearly specify if it's achieved or not.
– *Detection method:* Unavailable, as it depends on domain expertise and the goal model context.

4.16 Too specific element

– *Description:* Elements should describe the what, not the how, where, or when. This bad smell limits flexibility and may constrain the model's ability to adapt or address the broader goals effectively by focusing on particular implementation details.
– *Illustrative example:* An example of this bad smell is a goal stated as "Install new software updates every month" instead of "Ensure the system is up-to-date", which confines the model to a specific implementation detail rather than focusing on the broader objective of maintaining system currency.
– *Detection method:* Unavailable, as it depends on domain expertise and the goal model context.

4.17 Ambiguous element

– *Description:* Elements missing clarity and can be interpreted in more than one way.
– *Illustrative Examples:* "Go to the bank". Does *bank* refer to the financial institution or the river side?. The i* guide emphasized the use of "concise but informative phrases" [2]. Figure 10 [2] illustrates an example of an ambiguous element.
– *Detection method:* Unavailable, as it depends on domain expertise and the goal model context.

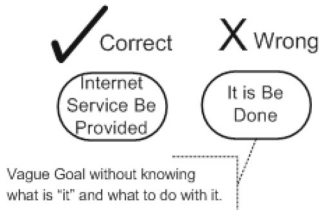


Fig. 10 Vague element [2]

5 Refactoring the linguistic bad smells in GRL models

In recent years, large language models (LLMs) have contributed to the further development of generative models, eliminating many of the problems associated with earlier models and introducing new capabilities. LLMs have the potential to eliminate problems such as limited vocabulary, grammatical errors, and lack of context understanding that were prevalent in earlier generative models. By leveraging vast amounts of training data, LLMs can generate more accurate and contextually relevant text, opening up new possibilities for the field of natural language generation. However, as of yet, there is no clear direction on how best to utilize LLMs for automated GRL linguistic refactoring.

This research makes use of GPT, an LLM developed by OpenAI ,[8] as it is one of the most capable LLMs for understanding languages at a human level [11, 41]. Specifically, we used the version "gpt−3.5-turbo". GPT's impressive capabilities go beyond just language understanding. With its large number of parameters, it can generate coherent and contextually relevant responses, engage in meaningful conversations, and perform tasks with minimal training [9]. This makes it a powerful tool for various applications, ranging from natural language processing to content generation. More specifically, we explore prompt engineering; a strategic and iterative process of designing prompts that generate desired responses from the GPT model. By tailoring prompts to target specific types of bad smell, we can optimize the efficacy of GPT for GRL refactoring.

We carefully followed OpenAI's prompt engineering guide and best practices to design the prompt. This involved selecting appropriate instructions, specifying the desired output format, and giving examples to the model to ensure accurate and reliable refactoring. We also iteratively refined the prompt based on the response to achieve optimal performance. When designing a prompt, there are three roles that can be used: System, User, and an optional Assistant role. The *System* role is responsible for providing the task and the instructions to the GPT model. These instructions can include asking the model to use a specific output format

```
prompt = [
  {
    "role": "system",
    "content": '''
        You are a specialist in English linguistics.
        <TASK>
        <CONSTRAINTS>
        Answer with the new sentence only.'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
  }
]
```

Fig. 11 Main prompts structure

or follow a particular set of guidelines. By carefully crafting the task and the instructions, researchers can guide the model's behavior and steer it toward generating the desired refactored sentence. The *User* role serves as the entity that interacts with GPT and asks questions. The *Assistant* role is particularly useful in the context of few-shot prompting and it is not in the scope of this research.

Following the prompt-engineering guidelines and iteratively refining the prompts, we arrived at the finalized main structure of the prompts, shown in Fig. 11. For the system role, we first instructed the model to act as a specialist in English linguistics. The reason for selecting this role is that a specialist in English linguistics possesses deep knowledge of syntax, semantics, and linguistic structures. This expertise is crucial for accurately analyzing and correcting sentence structures and meanings. Furthermore, unlike roles such as requirements engineers or software analysts, who may have specific domain knowledge but less focus on linguistic precision, a linguistics specialist is trained to evaluate and improve language quality independently of the content's context. In the next step, we explain the task to be performed by the model, along with any constraints that may apply, which are discussed later for each specific bad smells. Lastly, we provide instructions on the output format. For the user role, we provide the sentence to be refactored.

In some cases, GPT can demonstrate an extreme level of creativity, leading to substantial modifications in the sentence's structure. However, this may not always be desirable in certain applications, such as refactoring. To solve this issue, we utilize *temperature* parameter in GPT's API to control the level of randomness and creativity in the model's output during text generation. A higher value, such as 0.8, increases randomness and creativity, while a lower value, such as 0.2, results in more focused and deterministic responses. In this research, we set the temperature to 0.2 to limit the creativity.

[8] https://openai.com.

```
prompt = [
      {
        "role": "system",
        "content":
        '''You are a specialist in English linguistics.
          You will be provided with a sentence, and your
          task is to summarize it in <LENGTH_THRESHOLD>
          words or fewer.
          Comply with the following conditions:
          (1) Do not convert a verb phrase to a noun phrase,
          and vice versa.
          (2) Change as few words as possible.
          Answer with the new sentence only.'''
        },
      {
        "role": "user",
        "content": "<SENTENCE>"
        }
]
```

Fig. 12 Prompt for lengthy element refactoring

```
prompt = [
      {
        "role": "system",
        "content":
        '''You are a specialist in English linguistics.
        A complex sentence is a sentence with one independent
        clause and at least one dependent clause. A simple
        sentence has a single independent clause.
        You will be provided with a complex sentence, and your
        task is to make it a simple sentence.
        Do not convert a verb phrase to a noun phrase, and vice
        versa.
        Answer with the new sentence only.
        '''
        },
      {
        "role": "user",
        "content": "<SENTENCE>"
        }
]
```

Fig. 13 Prompt for complex element refactoring

In the following sections, the prompt-engineering process for each bad smell is discussed in more detail.

5.1 Lengthy element refactoring

Figure 12 shows the prompt used for lengthy element refactoring. To refactor the sentence, we asked the model to summarize it based on a length threshold value specified by the user. Since summarization requires creativity to some extent, the model might generate a sentence that contains another bad smell. To solve this issue, we added the constraints shown in Fig. 12.

5.2 Complex element refactoring

We started by testing GPT on its understanding of a complex element. In some cases, GPT responded with the correct definition of a complex element. In few cases, GPT answered with non-linguistic context, but rather about describing a concept that is complicated or has many aspects. We eliminated such cases by defining the complex element for GPT, and then stating the task. As previously stated, this task is subject to high creativity, so we add a constraint as in Fig. 13.

5.3 Punctuation-marked element refactoring

Figure 14 shows the refactoring prompt of the punctuation-marked element bad smell. This is a simple task that only requires removing the punctuation marks, hence, no additional instructions and constraints are required.

```
prompt = [
      {
        "role": "system",
        "content":
        ''' You are a specialist in English linguistics.
        You will be provided with a sentence, and your
        task is to remove all punctuation marks.
        Answer with the new sentence only.'''
        },
      {
        "role": "user",
        "content": "<SENTENCE>"
        }
]
```

Fig. 14 Prompt for Punctuation-marked element refactoring

```
prompt = [
      {
        "role": "system",
        "content":
        '''You are a specialist in English linguistics.
        You will be provided with a sentence that is a verb
        phrase, and your task is to make it a noun pharse
        representing an actor.
        Example of actors: System, PC User, and Privacy Officer.
        Answer with the new sentence only.'''
        },
      {
        "role": "user",
        "content": "<SENTENCE>"
        }
]
```

Fig. 15 Prompt for incorrect actor syntax refactoring

5.4 Incorrect actor syntax refactoring

Figure 15 shows the prompt used to refactor incorrect actor syntax. The task is to convert a verb phrase to a noun phrase. However, in some cases, GPT generates a correct noun phrase but does not represent an actor. For example, *provide health* as incorrect actor syntax was refactored to *health provision* which does not represent an actor. A correct refactoring would be *health provider*. To solve this issue, we provided examples for GPT to follow, as shown in Fig 15.

```
prompt = [
  {
    "role": "system",
    "content":
    '''
    You are a specialist in English linguistics.
    You will be provided with a sentence that is not
    a noun phrase, and your task is to make it a noun
    pharse representing a goal.
    For example: high data quality, fast response time,
    and course registration.
    Answer with the new sentence only.'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
  }
]
```

Fig. 16 Prompt for incorrect goal syntax refactoring

5.5 Incorrect goal syntax refactoring

Figure 16 shows the prompt used to refactor incorrect goal syntax. Similarly to refactoring of incorrect actor syntax in Sect. 5.4, the task is to convert a verb phrase to a noun phrase. However, we provided different examples to represent a goal.

It is important to note that we have not instructed the LLM to follow a specific style; the task is solely to convert a given phrase, intended to represent a goal, into a noun phrase without adhering to any specific stylistic guidelines.

5.6 Incorrect softgoal syntax refactoring

We use a similar prompt as refactoring incorrect goal syntax, described in Sect. 5.5.

5.7 Incorrect task syntax refactoring

Figure 17 shows the prompt used to refactor incorrect task syntax. The task is to convert a noun phrase to a verb phrase.

5.8 Incorrect resource syntax refactoring

Figure 18 shows the prompt used to refactor incorrect resource syntax. The task is to convert a verb phrase to a noun phrase. Additionally, we provided examples that represent a resource for GPT to follow.

5.9 Misspelled element refactoring

Misspelled words refactoring was implemented with the detection, as described in Sect. 4.9.

Please note that we have delegated the task of refactoring similar, mismatching, and conflicting elements to the user, allowing him to redefine his goals, soft goals, and tasks. This decision stems from the subjective nature of the task, recognizing that each user may have unique perspectives.

```
prompt = [
  {
    "role": "system",
    "content":
    '''
    You are a specialist in English linguistics.
    You will be provided with a sentence that is not
    a verb phrase, and your task is to make it a verb
    pharse representing a task.
    For example: provide maintenance services, help
    co-workers, and enhance quality.
    Answer with the new sentence only.'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
  }
]
```

Fig. 17 Prompt for incorrect task syntax refactoring

```
prompt = [
  {
    "role": "system",
    "content":
    '''
    You are a specialist in English linguistics.
    You will be provided with a sentence that is not
    a noun phrase, and your task is to make it a noun
    pharse representing a resource.
    For example: internet, database, and files system.
    Answer with the new sentence only.'''
  },
  {
    "role": "user",
    "content": "<SENTENCE>"
  }
]
```

Fig. 18 Prompt for incorrect resource syntax refactoring

6 Tool support

To automatically detect and refactor instances of the proposed bad smells, a dedicated and publicly available TGRL-based tool[9] has been developed. Moreover, the fine-tuned NLI model (described in Sect. 4) is publicly available.[10] The tool has been developed using Gradio Python library[11] and hosted by Hugging Face platform.[12]

Figure 19 illustrates the tool graphical user interface. The input to the tool is a TGRL model (.xgrl file), whereas

[9] https://huggingface.co/spaces/nouf-sst/TGRL-bad-smells.

[10] https://huggingface.co/nouf-sst/bert-base-MultiNLI.

[11] https://www.gradio.app.

[12] https://huggingface.co.

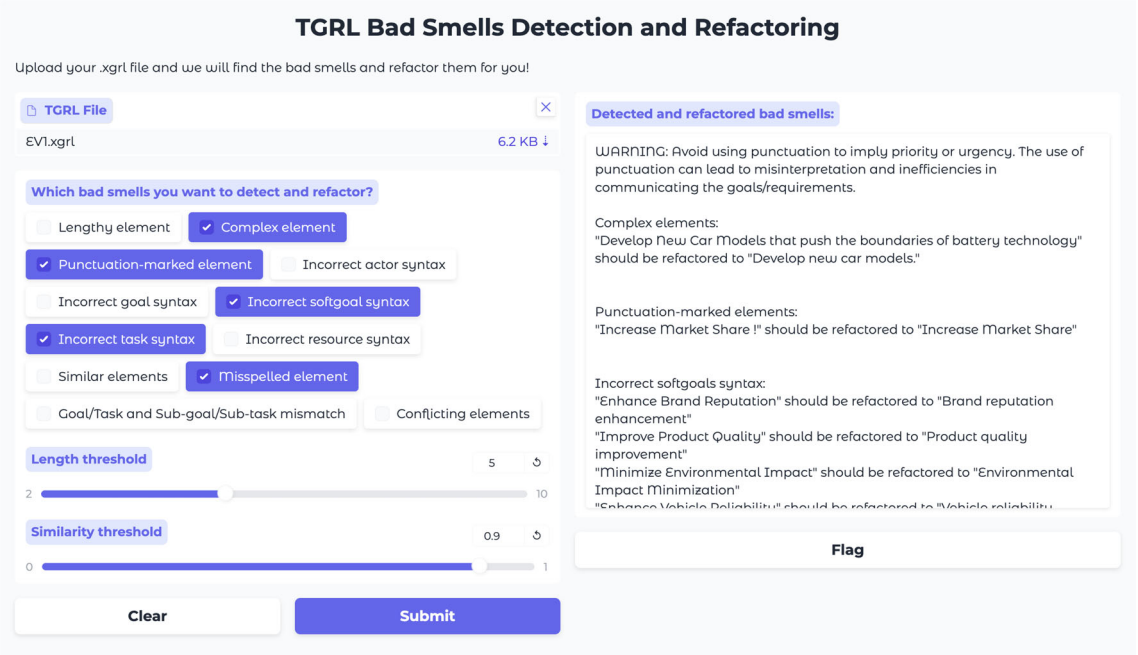


Fig. 19 A tool for TGRL linguistic bad smells detection and refactoring

the output is a list of detected bad smells and their refactoring. The tool starts by parsing the TGRL model (.xgrl) to extract all textual elements (i.e., actors, goals, softgoals, tasks, resources) and perform the needed pre-processing. The user has the option to select which bad smells to detect and refactor as well as what threshold values to use. Lastly, the tool lists the detected bad smells with their refactoring based on the user's choice.

7 Empirical evaluation

The aim of our empirical evaluation is twofold: (1) evaluate the enhancement we have proposed to the detection of the bad smells, and (2) evaluate the validity of the refactored intentional elements.

In Alturayeif and Hassine [3], we have evaluated the bad smells detection approach using two case studies: (1) University Alumni (adapted from Hassine and Amyot [19]) and (2) Healthcare system (originally introduced as an i* model by Sumesh et al. [46] and subsequently converted to GRL by Hassine and Tukur [20]). In Alturayeif and Hassine [3], three graduate students participated in the experiment. All three have previous experience with utilizing GRL models. We conducted introductory meetings with the participants to outline the assignment, provide definitions and examples of bad smells, and address any queries they had. Participants were provided with the two TGRL models and tasked with identifying any linguistic bad smells. We have analyzed manually the reported bad smells and compared them with the ones found by our automated tool. Additionally, if a participant overlooks a bad smell detected by the tool, he/she is prompted to confirm its presence and provide a rationale supporting his/her evaluation.

With the enhancements implemented in our detection tool, we have reexamined the data from the two case studies to evaluate the impact of these improvements (please refer to Sect. 7.4.1). However, this paper primarily concentrates on evaluating the refactoring of bad smells.

It is worth noting that the produced prompts were rigorously tested on a diverse set of sample sentences beyond those used in the empirical evaluation to avoid bias and overfitting, ensuring robust and generalizable results.

We follow the templates and recommendations presented by Wohlin et al. [50].

7.1 Subjects

In order to evaluate our proposed approach and tool, we have used four GRL models[13]:

- GRL model 1: University Alumni
- GRL model 2: Healthcare System
- GRL model 3: Electric Vehicle (illustrated in Fig. 1).
- GRL model 4: AI-Powered Mental Health Companion (illustrated in Fig. 23).

[13] https://github.com/Noufst/TGLR-Bad-Smells.

Models 1 and 2 are used to evaluate the effectiveness of enhancing the tool's detection capabilities compared to those presented in Alturayeif and Hassine [3]. Model 3 (Electric Vehicle) and Model 4 (AI-Powered Mental Health Companion) were developed by teams of undergraduate students enrolled in a requirements engineering course.

All four models are utilized to generate 71 instances of bad smells along with their corresponding refactoring solutions. The correctness of the refactoring is evaluated through a survey.[14]

7.2 Experimental task

The evaluation consists of 2 main tasks:

1. *Task 1: Assessment of the bad smells detection enhancements*: Apply the tool to models 1 and 2, noting the identified bad smells. Subsequently, compare these newly discovered bad smells with those previously identified by participants in the earlier evaluation [3].
2. *Task 2: Assessment of the correctness of the produced refactoring solutions*: Run the tool on all four models, then document all identified bad smells (71 bad smells) and their corresponding refactoring solutions. Next, we designed a survey comprising 72 questions. There are 71 questions (5-point Likert scale question to gauge the respondent's agreement or disagreement) pertaining to the correctness of the refactoring of the identified bad smells, and one open-ended question aimed at gathering additional insights or comments regarding the overall refactoring process. The correctness of the proposed refactoring solutions is rated from 1 to 5, with 1 being 'Strongly Disagree' (the refactoring is incorrect) and 5 being 'Strongly Agree' (the refactoring is correct).
   In addition, we have provided an online document that includes detailed descriptions of the targeted bad smells, along with illustrative examples, to ensure clarity and understanding.
   The online survey was sent to 21 potential participants, including 2 professors and 19 computer science graduate students. The eleven (19) graduate students who participated in the questionnaire had completed their undergraduate studies in computer science in English and are currently enrolled in a graduate program conducted in English, demonstrating their strong English proficiency. They had successfully completed a graduate-level course in requirements engineering, where they gained practical experience in goal-oriented modeling, with a particular focus on the goal-oriented requirements language (GRL). Their learning was reinforced through hands-on exercises, allowing them to apply theoretical concepts in a practical context, thereby deepening their understanding and proficiency in GRL. The two professors who completed the questionnaire have taught both graduate and undergraduate courses in requirements engineering. They possess extensive experience in goal-oriented modeling and are well-versed in GRL. In addition, it is important to note that while the participants have an interest in the topic, they remain independent and free from any bias, conflict of interest, or undue influence from the study's authors.

7.3 Effectiveness measurement

For the first experiment and in order to measure the effectiveness of our approach and tool, we use computed precision, recall, F1-score, and F2-score, metrics widely used in Information Retrieval (IR):

- *Precision*: To find how many of the identified bad smells are correct, using True Positives (TP) and False Positives (FP) (as in Eq. 1).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

- *Recall*: To find how many bad smells are identified correctly, using True Positives (TP) and False Negatives (FN) (as in Eq. 2).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

- *F1-score*: The harmonic mean of precision and recall, with equal weight to precision and recall (as in Eq. 3).

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- *F2-score*: The harmonic mean of precision and recall, with relatively greater weight to recall compared to precision (as in Eq. 4).

$$F2 = 5 \times \frac{\text{Precision} \times \text{Recall}}{4 \times \text{Precision} + \text{Recall}} \tag{4}$$

For the second task, we calculate descriptive statistics including the average, median, and quartiles for all questions per bad smell type. Additionally, we analyzed the open-ended responses.

7.4 Experiment results

In this section, we present and discuss the obtained experimental results.

[14] https://docs.google.com/forms/d/e/1FAIpQLScnSEVDx6y0ThnLAI_6PH_Q-O2JRal1IZfvAcrdS14f6SRghg/viewform.

Table 1 Detected bad smells by the proposed tool and the human evaluators (Model 1: University Alumni)

Bad smell type	Bad smell	Detected bad smells	
		Enhanced tool	By the human evaluators
Element Size	Use social media since students use it	✓	✓
	Provide access to the university facilities	✓	✓
Complex Sentences	Use social media since students use it	✓	✓
Punctuation Marks	Advertise job openings!	✓	✓
Incorrect goal syntax	Give back to University	✓	✓
Incorrect task syntax	Identify collaboration areas	✓	–
Similar Elements	Give back to University & Donate to the university	✓	–
	Helping current students & Mentor current students	✓	✓
	Establish research collaboration & Identify collaboration areas	✓	✓
	Effective Networking & Organize networking events	✓	–
	Enhanced collaboration with industry & Support industry related projects	✓	–
Goal/Task and Sub-goal/Sub-task Mismatch	Fostering the University Alumni relationship & Enhanced collaboration with industry	✓	–

Table 2 Detected bad smells by the proposed tool and the human evaluators (Model 2: Healthcare model)

Bad smell type	Bad smell	Detected bad smells	
		Enhanced tool	By the human evaluators
Incorrect actor syntax	Providing health	✓	✓
Incorrect goal syntax	Keep well	✓	✓
	Treat patients	✓	✓
Incorrect softgoal syntax	Reduce Expense	✓	✓
	Increase happiness	✓	✓
	Reduce maintenance cost	✓	✓
Incorrect task syntax	Patient-centered care	✓	✓
	Provider-centered care	✓	✓
Similar elements	Quality of care & Patient-centered care	✓	–
	Patient-centered care & Provider-centered care	✓	–
	Treat patients & Effective treatments	✓	–
	Treat patients & Patient-centered care	✓	–
Misspellings	Paitent	✓	✓
Goal/Task and sub-goal/sub-task mismatch	Keep well & Patient-centered care	✓	–
	Keep well & Provider-centered care	✓	–
	Treat patients & Patient-centered care	✓	–
	Treat patients & Provider-centered care	✓	–
Conflicting goals/tasks	Reduce expense & Increase happiness	✓	✓

7.4.1 Task 1 results

In Alturayeif and Hassine [3], reported a precision of 0.57, a recall of 0.95, an F1-score of 0.65, and an F2-score of 0.8. In this paper, we revisited the data from the two case studies to assess the impact of the improvements made to our detection tool. Tables 1 and 2 show the detected bad smells by the enhanced tool and the ones detected by the human evaluators for the University Alumni and the Healthcare models, respectively. Table 3 shows recall, precision, F1-score, and F2-score of each TGRL model and their overall performance. In our context, recall is the most important metric as it captures

Table 3 New evaluation results versus the results reported in Alturayeif and Hassine [3]

	Results reported in [3]			New results		
	University Alumni	Healthcare	Average	University Alumni	Healthcare	Average
Precision	0.44	0.55	0.50	0.58	0.57	0.575
Recall	1.00	0.92	0.95	1.00	1.00	1.00
F1-score	0.61	0.69	0.65	0.75	0.73	0.74
F2-score	0.80	0.80	0.80	0.88	0.87	0.875

how many of the actual bad smells are detected. However, precision cannot be totally ignored, as we do not want to simply provide the user with a large number of inaccurate bad smells. Therefore, F2-score is considered as a suitable metric since it accounts for both precision and recall, but gives more weight to recall. As shown in Table 3, the tool was able to achieve a precision of 0.575, a recall of 1, an F1-score of 0.74, and an F2-score of 0.875. These results outperform the ones obtained in Alturayeif and Hassine [3] (precision of 0.5, a recall of 0.95, an F1-score of 0.65, and an F2-score of 0.8).

7.4.2 Task 2 results

We present the results of the bad smells refactoring process in Tables 4, 5, 6, and 7, each corresponding to one of the four GRL models.

For the survey, we have collected a total of 13 responses, consisting of feedback from 2 professors and 11 graduate students.

For each bad smell type, we visualized the distribution of participants' ratings using a diverging stacked bar chart (Fig. 20). This representation provides a detailed view of how participants rated each bad smell type, categorizing responses into "1 Star", "2 Stars", "3 Stars", "4 Stars" and "5 Stars". This visualization highlights the distribution of opinions and the proportion of participants who found the refactored elements correct to varying degrees.

The overall ratings indicate strong agreement among participants, with a majority of responses falling into the "5 Stars" category. Refactored elements addressing "incorrect actor syntax" received the highest proportion of "5 Stars" ratings, indicating that participants perceived these sentences as highly correct. Similarly, "incorrect resource syntax", "misspelled element", and "incorrect task syntax" received positive ratings. Conversely, "complex elements" received relatively fewer "5 Stars" ratings and a higher proportion of "4 Stars" or "3 Stars" responses, suggesting participants perceived these refactored sentences as less correct compared to other types. The ambiguity and complexity present in complex elements can further complicate the refactoring process, making it a more challenging task compared to other tasks such as converting noun phrases to verb phrases.

The quartile analysis shows that most bad smells have high median values, indicating a general tendency toward positive ratings. Bad smells, like "Incorrect Actor Syntax", "Incorrect Resource Syntax" and "Misspelled Element", demonstrate no variability, with all quartiles equal to 5.00, suggesting clear agreement among participants. Conversely, "Complex Element" stands out with greater variability, as indicated by a relatively wide range between the first quartile (4.00) and the third quartile (5.00), reflecting more diverse responses. Overall, most bad smells exhibit high scores with minimal variability.

Further, there were a few participants who reported that the refactoring of "lengthy elements" was not sufficient and the resulting sentence was still lengthy. For example, the element "Develop new car models that push the boundaries of battery technology" was refactored into "Create innovative car models with advanced battery tech". Although the new element is still lengthy, the word count has been reduced from 11 to 8. Further reducing the element may negatively impact its meaning. Our tool, however, detects all lengthy elements 100% of the time. In some cases, users may need to evaluate the trade-off between having a short sentence and a complete meaningful sentence.

In summary, the high mean values suggest that participants perceived the refactored sentences as highly correct across the different types of linguistic bad smells. Additionally, the low variability suggest a general consensus among participants regarding the evaluation of the refactored sentences. The overall results indicate that the proposed approach effectively addresses various linguistic bad smells and improves the overall correctness of sentences.

8 Discussion

In this section, we introduce an integration framework that enables various goal modeling tools to interact with a server and request refactoring services, enhancing flexibility and reducing dependency on specific goal models. This framework improves the adaptability and generalizability of our approach across different goal modeling methods.

Figure 21 illustrates the integration framework, demonstrating how our approach can be applied to various goal

Table 4 Refactoring results (electric vehicle model)

Bad smell type		Original sentence	Refactored sentence
Lengthy element	1	Develop new car models that push the boundaries of battery technology	Create innovative car models with advanced battery tech
	2	Avoid exposure to high temperature environments and conditions	Stay away from extreme heat
	3	Exploring scenic routes or destinations suitable for EV travel	Finding EV-friendly scenic routes
Complex element	4	Develop new car models that push the boundaries of battery technology	Develop new car models
Punctuation-marked element	5	Increase market share !	Increase Market Share
Incorrect Goal Syntax	6	Increase market share !	Market share increase
	7	Reduce production costs	Cost reduction in production
	8	Expand product line	Expansion of product line
	9	Reduce manufacturing waste	Manufacturing waste reduction
	10	Achieve customer satisfaction	Customer Satisfaction Achievement
	11	Optimize charging schedule	Charging schedule optimization
	12	Extend battery lifespan	Battery lifespan extension
Incorrect Softgoal Syntax	13	Enhance brand reputation	Brand reputation enhancement
	14	Improve product quality	Product quality improvement
	15	Minimize environmental impact	Environmental Impact Minimization
	16	Enhance vehicle reliability	Vehicle reliability enhancement
	17	Meet safety regulations	Compliance with safety regulations
	18	Ensure safety compliance	Safety compliance
	19	Minimize operating costs	Operating cost minimization
	20	Maximize vehicle lifespan	Vehicle lifespan maximization
	21	Reduce charging expenses	Expense reduction
	22	Minimize maintenance costs	Cost minimization
Incorrect task syntax	23	Enhanced battery performance	Improve battery performance
	24	Frequent deep discharges	Deep discharge batteries frequently
	25	Exploring scenic routes or destinations suitable for EV travel	Explore scenic routes or destinations suitable for EV travel
Incorrect resource syntax	26	Manage EV fleet	EV fleet management
Misspelled element	27	Implement recicling programs	Implement recycling programs

models via a REST API. The figure shows the interaction between three main components: the Client, the API, and the Server, facilitating the detection and refactoring of bad smells in different goal models.

On the client side, the process starts with the user interface (GUI), where users can upload a goal model. This model is parsed and prepared for submission to the server using a model-specific parser. In this work, we developed a parser for TGRL to extract necessary elements, but the framework's flexibility allows for creating parsers for any goal model. These parsers produce key elements required by our approach, including 'all_elements' (all components within the model), 'elements_per_actor' (elements associated with each actor), 'decomposed_elements' (breakdown of complex elements), and 'contributing_elements_per_actor' (elements contributing to others within each actor). The client then constructs a JSON request containing the model elements and parameters guiding the refactoring process, such as bad smells to detect, length threshold for model elements, similarity threshold for identifying similar elements, and LLM settings, e.g., version and temperature. This request is forwarded to the server via the API, which acts as an intermediary.

On the server side, the request is handled by two core components: the detection approach and the refactoring approach proposed in this work. Once the processing is complete, the server prepares a response that is sent back to the client. The server's JSON response contains a status, the original model elements, and the newly refactored elements. Additionally, it includes details on the refactoring process, such as the

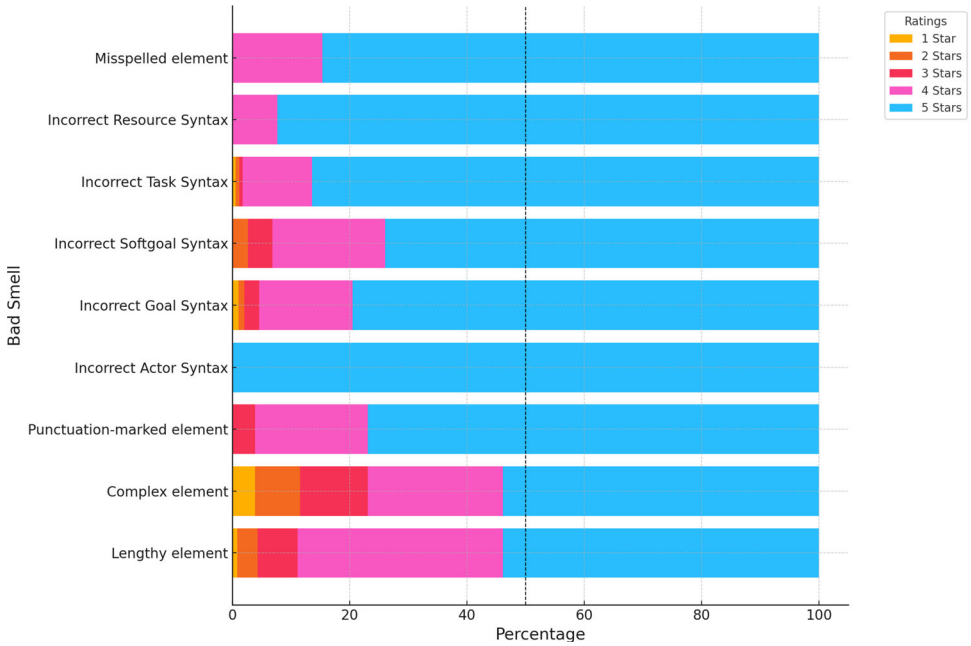


Fig. 20 Task 2 evaluation survey results

Fig. 21 An integration framework for goal models linguistic bad smells detection and refactoring

detected and undetected bad smells, along with the specific parameters used.

9 Threats to validity

Our approach, the tool, the examples and the empirical evaluation are subjected to several limitations and threats to validity that we categorize according to three important types of threats identified by Wright et al. [51].

9.1 Internal validity

A potential risk is that some participants may not have answered the questions seriously, which could be addressed in future evaluations by implementing measures to ensure greater participant engagement and accountability. Additionally, there is a potential risk of bias in the selection of GRL models for the empirical evaluation. The selected GRL models may have been particularly favorable to the technique, meaning the strong performance observed could be attributed more to the choice of models than to the technique itself. This introduces a potential bias that could distort the cause-and-effect relationship between the technique and its results. To partially mitigate this risk, we utilized two pre-existing publicly available GRL models, i.e., the University Alumni and the Healthcare models. Another possible threat to internal validity is that the empirical evaluation was conducted by participants who may not have had uniform levels of proficiency in the GRL language and English linguistic capabilities. This variability could potentially affect the results, as differences in language proficiency might influence how participants understand and apply the technique. To mitigate this risk, we selected subjects with comparable expertise in GRL modeling and provided them with uniform documentation of the targeted bad smells. Additionally, we believe that any differences in English proficiency had minimal impact due to the simplicity of the text and the use of commonly understood vocabulary. However, the potential for variability in participant proficiency remains a concern that could influence the internal validity of the findings.

9.2 Construct validity

A potential threat to construct validity is the inadequate representation of linguistic anomalies in the empirical validation. The goal models used in the study might not encompass the full spectrum of linguistic anomalies that the prompts are designed to address. For example, Noun phrases (NPs) and verb phrases (VPs) exhibit a range of structures. Noun phrases can be simple, with just a noun or pronoun, or complex, including determiners and modifiers. They may also feature prepositional phrases, infinitive phrases, or relative clauses. Verb phrases can be simple or complex, incorporating objects and adverbials. They can be transitive, with a direct object, intransitive, without an object, or ditransitive, with both direct and indirect objects. Additionally, verb phrases may include phrasal verbs or modal verbs. We did not cover all possible variations of linguistic anomalies, which compromises the construct validity of our evaluation.

9.3 External validity

One potential concern is that the applicability of our approach was demonstrated using only four GRL models. Although these models encompass all types of linguistic bad smells, it is essential to include additional real-world examples to more comprehensively assess the effectiveness and generalizability of our approach. A related external validity concern stems from the fact that Model 1 was developed by the second author, while Models 3 and 4 were created by undergraduate student teams under the guidance of the same author. As a result, the author's writing style and philosophy may have influenced the way elements in these models were constructed. Although these models were selected for their accessibility and the authors' detailed familiarity with their development, future research should incorporate independent goal models from the literature to enhance generalizability. Another possible threat is related to the selection of the evaluators. Although the evaluators were not taught directly by the authors of this study, they completed the graduate requirements engineering course at the same institution, where professors generally share a similar modeling culture to that of the authors. Another potential threat to external validity is that our linguistic bad smells detection and refactoring technique is currently tailored to TGRL [1]. However, we are confident that the approach can be readily adapted to encompass other languages, such as i* [52], which share similar constructs. Additionally, we introduced an integration framework (Sect. 8) to demonstrate how our proposed approach can be integrated and generalized to other goal modeling methods. Yet another external validity threat relates to the (medium) sizes of the four models used. The observed benefits of our approach may not be different on very large models since we treat each intentional element separately. The benefits observed from our approach may remain consistent even with very large models, as we address each intentional element individually. Finally, a potential issue regarding external validity arises from the dependency on GPT for both detection and refactoring tasks. Different Language Model Models (LLMs) exhibit distinct capabilities and limitations, often requiring customized prompts for optimal performance. Therefore, it is essential to explore alternative versions and models of LLMs to address this limitation effectively.

10 Conclusion and future work

Goal models are susceptible to various linguistic bad smells, including incorrect goal statements, conflicting intentional elements, and spelling errors, which can impede effective communication and lead to misunderstandings and inconsistencies. To ensure the quality and accuracy of goal models, it is essential to identify and address these linguistic bad smells. In this study, we expand upon prior research by enhancing the catalog of 17 linguistic bad smells in goal-oriented requirements languages (GRL), refining detection techniques using a combination of NLP-based and LLM-based methods, thereby significantly improving detection capabilities. Additionally, we provide automated refactoring solutions for 9 of these bad smells using GPT prompts, leaving the remaining four for user discretion due to their subjective nature. These detection and refactoring processes are implemented in a tool tailored to the Textual GRL (TGRL) language. Evaluation of the approach and tool involves administering a questionnaire to 13 participants, who assessed the correctness of the refactoring of 71 linguistic bad smells identified in four TGRL models. Participants perceived the refactored sentences as highly correct across the different types of linguistic bad smells.

As future work, we plan to broaden the evaluation scope by integrating industrial GRL models, thereby enhancing the robustness and applicability of our findings. Another potential future research direction consists of proposing techniques to detect and refactor the remaining five bad smells that have not yet been addressed. Finally, an intriguing future direction is to extend the proposed LLM-based linguistic refactoring technique to repair other types of models beyond its current application. This approach has the potential to address linguistic inconsistencies in a wide variety of models, including those used in software design (e.g., UML diagrams), requirements engineering, and formal specification.

A TGRL specification of the electric vehicle model (Electric Car Owner)

See Fig. 22.

```
    actor ElectricCarOwner {

        name = "Electric Car Owner";

        // ------- Softgoals -------
        softGoal GoodVehicleReliability {
            name = "Good Vehicle Reliability";
            importance = high;
        }
        softGoal MinimizeOperatingCosts {
            name = "Minimize Operating Costs";
            importance = 75;
        }
        softGoal MaximizeVehicleLifespan {
            name = "Maximize Vehicle Lifespan";
            decompositionType = and;
        }
        softGoal ReduceChargingExpenses {
            name = "Reduce Charging Expenses";
        }
        softGoal MinimizeMaintenanceCosts {
            name = "Minimize Maintenance Costs";
        }

        // ------- Goals -------
        goal OptimizeChargingSchedule {
            name = "Optimize Charging Schedule";
            importance = high;
        }
        goal ExtendBatteryLifespan {
            name = "Extend Battery Lifespan";
            decompositionType = or;
        }

        // ------- Tasks -------
        task DriveResponsibly {
            name = "Drive Responsibly";
        }
        task FollowManufacturersMaintenanceSchedule {
            name = "Follow Manufacturer Maintenance Schedule";
        }
        task AvoidExposuretoHighTemperature {
            name = "Avoid Exposure to High Temperature Environments and Conditions";
        }
        task FrequentDeepDischarges {
            name = "Frequent Deep Discharges";
        }
        task UtilizeOffPeakCharging {
            name = "Utilize Off-Peak Charging";
        }
        task ImplementPredictiveMaintenance {
            name = "Implement Predictive Maintenance";
        }
        task ExploringScenicRoutes {
            name = "Exploring scenic routes or destinations suitable for EV Travel";
        }
        // ------ Decomposition -------
        MaximizeVehicleLifespan decomposedBy DriveResponsibly, FollowManufacturersMaintenanceSchedule;

        // ------ Contributions -------
        MaximizeVehicleLifespan contributesTo GoodVehicleReliability {50;};
        ExtendBatteryLifespan contributesTo GoodVehicleReliability {50;};
        AvoidExposuretoHighTemperature contributesTo ExtendBatteryLifespan {25;};
        FrequentDeepDischarges contributesTo ExtendBatteryLifespan {-25;};
        DriveResponsibly contributesTo ExtendBatteryLifespan {25;};
        OptimizeChargingSchedule contributesTo ExtendBatteryLifespan {25;};
        OptimizeChargingSchedule contributesTo ReduceChargingExpenses {25;};
        UtilizeOffPeakCharging contributesTo ReduceChargingExpenses {25;};
        ImplementPredictiveMaintenance contributesTo MinimizeMaintenanceCosts {75;};
        MinimizeMaintenanceCosts contributesTo MinimizeOperatingCosts {25;};
        ReduceChargingExpenses contributesTo MinimizeOperatingCosts {25;};
        ExploringScenicRoutes contributesTo MinimizeMaintenanceCosts {25;};

        // ------ Dependencies -------
        GoodVehicleReliability dependsOn ElectricCarManufacturer.EnhanceVehicleReliability;
    }

    resource ManageEVFleet {
        name = "Manage EV Fleet";
        }

}
```

Fig. 22 TGRL specification of the electric vehicle model (Electric Car Owner)

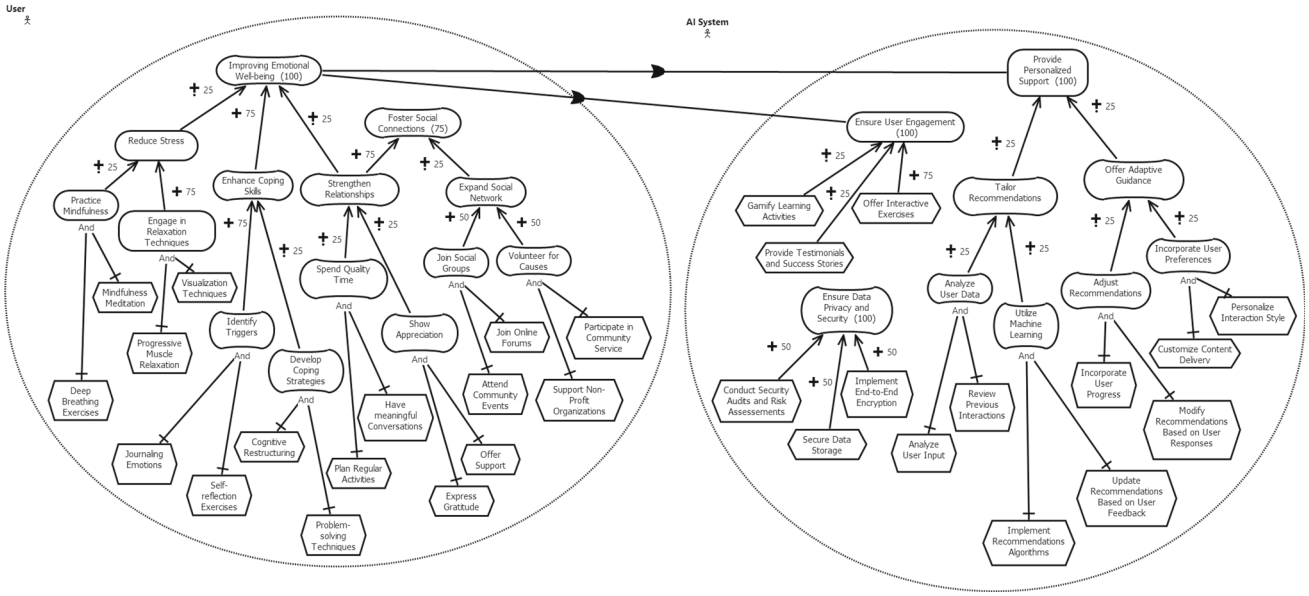


Fig. 23 AI-powered mental health companion GRL model

B GRL model of the AI-powered mental health companion

See Fig. 23.

C Refactoring output

See Tables 5, 6, 7.

Table 5 Refactoring results (University Alumni Model)

Bad smell type		Original sentence	Refactored Sentence
Lengthy elements	1	Use social media since students use it	Students use social media
	2	Provide access to the university facilities	Grant access to university facilities
Complex elements	3	Use social media since students use it	Students use social media
Punctuation-marked elements	4	Advertise job openings!	Advertise job openings
Incorrect goal syntax	5	Give back to University	Contribution to the University

Table 6 Refactoring results (healthcare model)

Bad smell type		Original sentence	Refactored sentence
Incorrect actor syntax	1	Providing health	Health provider
Incorrect goal syntax	2	Keep well	Well-being
	3	Treat Patients	Patient treatment
Incorrect Softgoal Syntax	4	Reduce expense	Expense reduction
	5	Increase happiness	Achieving increased happiness
	6	Reduce maintenance cost	Cost reduction in maintenance
Incorrect Task Syntax	7	Patient-centered care	Provide patient-centered care
	8	Provider-centered care	Provide provider-centered care
Misspelled element	9	Paitent	Patient

Table 7 Refactoring results (AI-powered mental health companion model)

Bad smell type		Original sentence	Refactored sentence
Lengthy element	1	Conduct security audits and risk assessements	Perform security audits and assessments
	2	Update recommendations based on user feedback	Revise recommendations from user input
	3	Modify recommendations based on user responses	Revise suggestions from user feedback
Incorrect Goal Syntax	4	Reduce stress	Stress Reduction
	5	Practice mindfulness	Mindfulness practice
	6	Engage in relaxation techniques	Relaxation techniques engagement
	7	Provide personalized support	Supporting individuals with personalized assistance
	8	Ensure user engagement	User engagement
Incorrect Softgoal Syntax	9	Foster social connections	Social Connection Building
	10	Enhance coping skills	Improved coping skills
	11	Develop coping strategies	Coping strategies development
	12	Strengthen relationships	Relationship Strengthening
	13	Expand social network	Expansion of social network
	14	Spend quality time	Quality Time Spent
	15	Show appreciation	Appreciating others
	16	Join social groups	Social Group Membership
	17	Ensure data privacy and security	Data privacy and security
	18	Analyze user data	User data analysis
	19	Utilize machine learning	Machine learning utilization
	20	Offer adaptive guidance	Adaptive guidance offering
	21	Adjust recommendations	Improved recommendation system
	22	Incorporate user preferences	User preference incorporation
Incorrect Task Syntax	23	Deep breathing exercises	Practice deep breathing exercises
	24	Mindfulness meditation	Practice mindfulness meditation
	25	Progressive muscle relaxation	Practice progressive muscle relaxation
	26	Visualization techniques	Use visualization techniques
	27	Self-reflection exercises	Practice self-reflection exercises
	28	Cognitive restructuring	Practice cognitive restructuring
	29	Problem-solving techniques	Use problem-solving techniques
	30	Secure data storage practices	Practice secure data storage
Misspelled element	31	Conduct security audits and risk assessements	Conduct security audits and risk assessments

References

1. Abdelzad, V., Amyot, D., Lethbridge, T.C.: Adding a textual syntax to an existing graphical modeling language: Experience report with GRL. In: J. Fischer, M. Scheidgen, I. Schieferdecker, R. Reed (eds.) SDL 2015: Model-Driven Engineering for Smart Cities - 17th International SDL Forum, Berlin, Germany, October 12-14, 2015, Proceedings, *LNCS*, 9369, 159–174. Springer (2015). https://doi.org/10.1007/978-3-319-24912-4_12
2. Abdulhadi, S., Horkoff, J., Yu, E., Grau, G.: i* guide. http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guide&structure=i%2A+Guide. Last Accessed March (2024)
3. Alturayeif, N., Hassine, J.: Detection of linguistic bad smells in GRL models: An NLP approach. In: ACM/IEEE international conference on model driven engineering languages and systems, MODELS 2023 Companion, Västerås, Sweden, October 1–6, 2023, pp. 318–327. IEEE (2023). https://doi.org/10.1109/MODELS-C59198.2023.00062
4. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.S.K.: Evaluating goal models within the goal-oriented requirement language. Int. J. Intell. Syst. **25**(8), 841–877 (2010). https://doi.org/10.1002/int.20433
5. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A lightweight GRL profile for i* modeling. In: C.A. Heuser, G. Pernul (eds.) Advances in Conceptual Modeling—Challenging Perspectives, ER 2009 Workshops, Gramado, Brazil, November 9-12, 2009. Proceedings, *LNCS*, 5833, 254–264. Springer (2009). https://doi.org/10.1007/978-3-642-04947-7_31
6. Arnaoudova, V., Penta, M.D., Antoniol, G.: Linguistic antipatterns: what they are and how developers perceive them. Empir. Softw. Eng. **21**(1), 104–158 (2016). https://doi.org/10.1007/s10664-014-9350-8

7. Asano, K., Hayashi, S., Saeki, M.: Detecting bad smells of refinement in goal-oriented requirements analysis. In: S. de Cesare, U. Frank (eds.) Advances in Conceptual Modeling—ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings, *LNCS*, 10651, 122–132. Springer (2017). https://doi.org/10.1007/978-3-319-70625-2_12
8. Barriga, A., Heldal, R., Rutle, A., Iovino, L.: PARMOREL: a framework for customizable model repair. Softw. Syst. Model. **21**(5), 1739–1762 (2022). https://doi.org/10.1007/s10270-022-01005-0
9. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)
10. Cares, C., Franch, X.: Towards a framework for improving goal-oriented requirement models quality. In: C.P. Ayala, C.T.L.L. Silva, H. Astudillo (eds.) Anais do WER09—Workshop em Engenharia de Requisitos, Valparaíso, Chile, Julho 16-17, 2009 (2009). http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER09/cares.pdf
11. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: scaling language modeling with pathways. J. Mach. Learn. Res. **24**(240), 1–113 (2023)
12. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. In International Series in Software Engineering, vol. 5. Springer (2000). https://doi.org/10.1007/978-1-4615-5269-7
13. Espada, P., Goulão, M., Araújo, J.: A framework to evaluate complexity and completeness of KAOS goal models. In: C. Salinesi, M.C. Norrie, O. Pastor (eds.) Advanced Information Systems Engineering—25th International Conference, CAiSE 2013, Valencia, Spain, June 17–21, 2013. Proceedings, *Lecture Notes in Computer Science*, 7908, 562–577. Springer (2013). https://doi.org/10.1007/978-3-642-38709-8_36
14. Fowler, M.: Refactoring—Improving the design of existing code, 2nd edition. Addison Wesley object technology series. Addison-Wesley (2018). http://martinfowler.com/books/refactoring.html
15. Frantiska, J.: Use Case Diagrams, pp. 1–8. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-67440-7_1
16. Garnier, M., Saint-Dizier, P.: Error typology and remediation strategies for requirements written in english by non-native speakers. In: N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (eds.) Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016. European Language Resources Association (ELRA) (2016). http://www.lrec-conf.org/proceedings/lrec2016/summaries/80.html
17. Gralha, C., Araújo, J., Goulão, M.: Metrics for measuring complexity and completeness for social goal models. Inf. Syst. **53**, 346–362 (2015). https://doi.org/10.1016/j.is.2015.03.006
18. Gralha, C., Goulão, M., Araújo, J.: Identifying modularity improvement opportunities in goal-oriented requirements models. In: M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, J. Horkoff (eds.) Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16–20, 2014. Proceedings, *Lecture Notes in Computer Science*, vol. 8484, pp. 91–104. Springer (2014). https://doi.org/10.1007/978-3-319-07881-6_7
19. Hassine, J., Amyot, D.: A questionnaire-based survey methodology for systematically validating goal-oriented models. Requir. Eng. **21**(2), 285–308 (2016). https://doi.org/10.1007/s00766-015-0221-7
20. Hassine, J., Tukur, M.: Measurement and classification of interactor dependencies in goal models. Softw. Syst. Model. **21**(6), 2267–2310 (2022). https://doi.org/10.1007/s10270-021-00961-3
21. Horkoff, J., Aydemir, F.B., Cardoso, E., Li, T., Maté, A., Paja, E., Salnitri, M., Piras, L., Mylopoulos, J., Giorgini, P.: Goal-oriented requirements engineering: an extended systematic mapping study. Requir. Eng. **24**(2), 133–160 (2019). https://doi.org/10.1007/s00766-017-0280-z
22. ITU-T: Recommendation Z.151 (10/18), User Requirements Notation (URN) language definition, Geneva, Switzerland (2018). http://www.itu.int/rec/T-REC-Z.151/en
23. jUCMNav: v7.0.0. https://github.com/JUCMNAV, University of Ottawa, Canada. Last Accessed March 2024
24. Kaiya, H., Horai, H., Saeki, M.: AGORA: attributed goal-oriented requirements analysis method. In: 10th Anniversary IEEE Joint international conference on requirements engineering (RE 2002), 9-13 September 2002, Essen, Germany, pp. 13–22. IEEE Computer Society (2002). https://doi.org/10.1109/ICRE.2002.1048501
25. Kamata, M.I., Tamai, T.: How does requirements quality relate to project success or failure? In: 15th IEEE international requirements engineering conference, RE 2007, October 15-19th, 2007, New Delhi, India, pp. 69–78. IEEE Computer Society (2007). https://doi.org/10.1109/RE.2007.31
26. Knauss, E., Boustani, C.E.: Assessing the quality of software requirements specifications. In: 16th IEEE International requirements engineering conference, RE 2008, 8-12 September 2008, Barcelona, Catalunya, Spain, pp. 341–342. IEEE Computer Society (2008). https://doi.org/10.1109/RE.2008.29
27. Knauss, E., Boustani, C.E., Flohr, T.: Investigating the impact of software requirements specification quality on project success. In: F. Bomarius, M. Oivo, P. Jaring, P. Abrahamsson (eds.) Product-Focused Software Process Improvement, 10th International Conference, PROFES 2009, Oulu, Finland, June 15–17, 2009. Proceedings, Lecture Notes in Business Information Processing, 32, 28–42. Springer (2009). https://doi.org/10.1007/978-3-642-02152-7_4
28. Kravchenko, T., Bogdanova, T., Shevgunov, T.: Ranking requirements using moscow methodology in practice. In: Silhavy, R. (ed.) Cybernetics Perspectives in Systems, pp. 188–199. Springer International Publishing, Cham (2022)
29. Krogstie, J., Lindland, O.I., Sindre, G.: Towards a deeper understanding of quality in requirements engineering. In: J.A.B. Jr., J. Krogstie, O. Pastor, B. Pernici, C. Rolland, A. Sølvberg (eds.) Seminal contributions to information systems engineering, 25 Years of CAiSE, pp. 89–102. Springer (2013). https://doi.org/10.1007/978-3-642-36926-1_7
30. Marchezan, L., Kretschmer, R., Assunção, W.K.G., Reder, A., Egyed, A.: Generating repairs for inconsistent models. Softw. Syst. Model. **22**(1), 297–329 (2023). https://doi.org/10.1007/s10270-022-00996-0
31. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Comput. Linguistics **19**(2), 313–330 (1993)
32. May, P.: Machine translated multilingual sts benchmark dataset (2021). https://github.com/PhilipMay/stsb-multi-mt
33. Mohammed, M.A., Alshayeb, M.R., Hassine, J.: A search-based approach for detecting circular dependency bad smell in goal-oriented models. Softw. Syst. Model. **21**(5), 2007–2037 (2022). https://doi.org/10.1007/s10270-021-00965-z
34. Mohammed, M.A., Alshayeb, M.R., Hassine, J.: A rule-based approach for the identification of quality improvement opportunities in GRL models. Softw. Qual. J. **32**(3), 1007–1037 (2024). https://doi.org/10.1007/s11219-024-09679-z
35. Mohammed, M.A., Hassine, J., Alshayeb, M.R.: GSDetector: a tool for automatic detection of bad smells in GRL goal models. Int. J.

Softw. Tools Technol. Transf. **24**(6), 889–910 (2022). https://doi.org/10.1007/s10009-022-00662-2
36. Moody, D.L., Heymans, P., Matulevicius, R.: Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requir. Eng. **15**(2), 141–175 (2010). https://doi.org/10.1007/s00766-010-0100-1
37. Muhammad, T., Omar, S., Hassine, J.: Requirement engineering challenges: a systematic mapping study on the academic and the industrial perspective. Arab. J. Sci. Eng. **46**, 3723–3748 (2021). https://doi.org/10.1007/s13369-020-05159-1
38. Mussbacher, G., Amyot, D., Heymans, P.: Eight deadly sins of GRL. In: J.B. de Castro, X. Franch, J. Mylopoulos, E.S.K. Yu (eds.) Proceedings of the 5th International *i** Workshop 2011, Trento, Italy, August 28-29, 2011, CEUR Workshop Proceedings, 766, 2–7. CEUR-WS.org (2011). https://ceur-ws.org/Vol-766/paper01.pdf
39. Pohl, K.: Requirements Engineering - Fundamentals, Principles, and Techniques. Springer (2010). http://www.springer.com/computer/swe/book/978-3-642-12577-5?changeHeader
40. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A python natural language processing toolkit for many human languages. arXiv preprint arXiv:2003.07082 (2020)
41. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
42. Rago, A., Marcos, C.A., Diaz-Pace, J.A.: Identifying duplicate functionality in textual use cases by aligning semantic actions. Softw. Syst. Model. **15**(2), 579–603 (2016). https://doi.org/10.1007/s10270-014-0431-3
43. Sajjad, H., Durrani, N., Dalvi, F., Alam, F., Khan, A.R., Xu, J.: Analyzing encoded concepts in transformer language models. In: M. Carpuat, M. de Marneffe, I.V.M. Ruíz (eds.) Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022, pp. 3082–3101. Association for Computational Linguistics (2022). https://doi.org/10.18653/v1/2022.naacl-main.225
44. Santos, M., Gralha, C., Goulão, M., Araújo, J., Moreira, A., Cambeiro, J.: What is the impact of bad layout in the understandability of social goal models? In: 24th IEEE International requirements engineering conference, RE 2016, Beijing, China, September 12-16, 2016, pp. 206–215. IEEE Computer Society (2016). https://doi.org/10.1109/RE.2016.51
45. Seki, Y., Hayashi, S., Saeki, M.: Detecting bad smells in use case descriptions. In: D.E. Damian, A. Perini, S. Lee (eds.) 27th IEEE international requirements engineering conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019, pp. 98–108. IEEE (2019). https://doi.org/10.1109/RE.2019.00021
46. Sumesh, S., Krishna, A., Subramanian, C.M.: Game theory-based reasoning of opposing non-functional requirements using interactor dependencies. Comput. J. **62**(11), 1557–1583 (2019). https://doi.org/10.1093/comjnl/bxy143
47. Taylor, A., Marcus, M., Santorini, B.: The penn treebank: an overview. Treebanks: Building and using parsed corpora pp. 5–22 (2003)
48. Teruel, M.A., Navarro, E., López-Jaquero, V., Simarro, F.M., González, P.: CSRML: A goal-oriented approach to model requirements for collaborative systems. In: M.A. Jeusfeld, L.M.L. Delcambre, T.W. Ling (eds.) Conceptual Modeling—ER 2011, 30th International Conference, ER 2011, Brussels, Belgium, October 31—November 3, 2011. Proceedings, Lecture Notes in Computer Science, vol. 6998, pp. 33–46. Springer (2011). https://doi.org/10.1007/978-3-642-24606-7_4
49. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426 (2017)
50. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B.: Experimentation in Software Engineering. Springer (2012). https://doi.org/10.1007/978-3-642-29044-2
51. Wright, H.K., Kim, M., Perry, D.E.: Validity concerns in software engineering research. In: G. Roman, K.J. Sullivan (eds.) Proceedings of the Workshop on Future of Software Engineering Research, FoSER 2010, at the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010, pp. 411–414. ACM (2010). https://doi.org/10.1145/1882362.1882446
52. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, RE'97, pp. 226–235. IEEE Computer Society (1997). https://doi.org/10.1109/ISRE.1997.566873

Nouf Alturayeif is a Ph.D. candidate at King Fahd University of Petroleum and Minerals, specializing in the interdisciplinary application of Artificial Intelligence (AI)-including Natural Language Processing (NLP), Machine Learning (ML), and Large Language Models (LLMs) to address challenges in software engineering. She is a lecturer in the Computing Department at Imam Abdulrahman Bin Faisal University. Nouf has authored several impactful articles published in high-impact journals, including Automated Software Engineering and Expert Systems with Applications. Her research aims to advance software engineering practices by leveraging innovative AI-driven solutions.

Jameleddine Hassine is an Associate Professor at the department of Information and Computer Science of King Fahd University of Petroleum and Minerals (KFUPM). Dr. Hassine holds a Ph.D. from Concordia University, Canada (2008) and an M.Sc. from the University of Ottawa, Canada (2001). Dr. Hassine has several years of industrial experience within worldwide telecommunication companies; Nortel Networks (Canada) and Cisco Systems (Canada). His main research interests include requirements engineering (languages and methods), software testing, formal methods, software evaluation, and maintenance. He is actively involved in several funded research projects and has over 60 publications on various research topics in his field. Dr. Hassine published his research in many high-impact journals like Requirements Engineering Journal (REJ), Journal of Systems and Software (JSS), Information and Software Technology (IST), and Software and Systems Modeling (SoSyM).