Towards Automated Goal Model Generation from UML Use Case and Swimlane Diagrams

Ahlem Yousef*, Said Ghoul*, and Mohammad Tayae*
Philadelphia University, Amman, Jordan

Abstract

The Goal Model of software is one of the important concepts in the goal-based requirements engineering. It helps in specifying the software goals and the relationships between them. Several research works were conducted to generate Goal Model of software from its requirements documents. However, the generated Goal Models merge behavior and soft goals into a single model unit. This merging leads to tangled and complex generated Goal Models. Therefore, the maintenance of these models is hard and costly. The work presented in this paper proposes an approach splitting the generated Goal Model into three separated concerns (aspects) models (behavior, soft, and constraints) that facilitate its evolution and maintenance. The proposed approach is semi-automated, taking UML use case and Swimlane diagrams as inputs and generating a separated aspects model GM as output. The separation of Goal Model aspects led to adding new required information in input requirements specification documents. The feasibility of the proposed approach was validated on a concrete business case (Philadelphia University Quality Assurance Agenda). Its implementation was demonstrated through processes programming with algorithms and UML. Its contribution was demonstrated through its comparison with similar works. According to the observed results, this approach could be valuable in any goal-oriented requirements engineering application.

Key Words: Goal model (GM), behavior goal, soft goal, unified modeling language (UML), UML use case diagram, UML swim lane diagram, goal model generation, goal model maintenance, separation of concerns, aspects programming.

1 Introduction

Software requirements analysis is an approach that allows a better understanding of the requirements collected from stakeholders [3, 17, 19] as it is stated in requirements engineering [12, 20, 22]. These requirements are often complex and extensive [10].

One of the most important concepts in the software requirements analysis and specification is the Goal Model (GM) [1, 4, 9, 11]. IT helps in the definition of a collection of software goals as well as their relationships. It is one of the most important topics in software requirements and specifications analysis [5].

The approaches to generating the GMs differ. Some of them adopted the question method [14] and some others used parsing tree [21]. They also used different forms of requirements documents (textual documents and UML diagrams). The generated goal is modeled with the basic notations of GMs. All the current approaches generate a one piece tangled model merging soft and behavior goals [7, 8, 13, 14, 15, 18, 21, 24]. This tangling leads to problems in maintenance (bad quality and high cost) [6, 8].

As a solution to the problem raised up by the one-piece tangled GM, generated by the current approaches, this paper proposes a semi-automated approach that generates a GM from the requirements specification documents: UML use case and Swimlane diagrams. The generated GM separates clearly the aspects [16, 20] of behavior goals, soft goals, and the relationships between them. So, the obtained GM is composed of separated behavior GM, soft GM, and constraints that specify the relationships between the two separated models. This separation of aspects required the addition of some basic information in the input UML use case and Swimlane.

2 Background

2.1 Goal

GMs are elements of requirements engineering that may also be used more widely in business analysis. Related elements include stakeholder analysis, context analysis, and scenarios [2] among other business and technical areas. Actors' goals are visualized within the boundaries of the actor's goals, tasks are linked through links, and many dependencies such as quality and resources can be represented [7]. GMs are based on the following concepts and relationships between them: goal, task, role, quality, resource, actor, and actor boundary. Some of these concepts concern soft goals (such as: task, actor, actor boundary, resource, quality, ...). Others concern behavior goals (such as: goal).

A goal is the result toward which effort is directed to achieve this very result or objective. Goals are most commonly expressed as imperative sentences beginning with a verb (as in the examples below). For example: "Ensure that only the

* Research Laboratory on Bioinspired Software Engineering. Email: Ahloom13@gmail.com, shgoul@philadelphia.edu.jo, mtaye@philadelphia.edu.jo

account owner can edit his account details", "Allow admin to manage all accounts' privileges".

2.2 Types of Goal

Goals are usually classified in different categories related to their function, behavior, kind, or temporal characteristics:

- *Functional / Non-functional:* Functional goals express services that a system has to deliver. All other goals are non-functional including goals related to the "-ilities" (suitability, reliability, usability, interoperability, verifiability, ...).
- *Behavioral / Developmental-quality:* Behavioral goals express what a system's behavior is to be. These goals are satisfied (or not) by what the system does when it runs. Developmental quality goals express the process by which a system is produced and evolved. These goals are satisfied (or not) by the actions of the people responsible for producing a system.
- *Hard / soft:* Some goals are either satisfied or not satisfied, there is no in-between. Other goals cannot be completely satisfied, but only satisfied to a degree; these are called soft goals. A soft goal is satisfied if it is achieved to a degree that is acceptable, with the understanding that this may cover a wide range of relative achievement, and that complete achievement is not possible. If a soft goal is not satisfied, then it is denied.
- *Achieve / Maintain / Avoid/ Optimize:* An achieve goal refers to a property that is not initially true, but that becomes true, i.e., in terms of temporal logic: *CurrentCondition => eventually TargetCondition*

A maintain goal refers to a property that starts out true and stays that way, i.e., in terms of temporal logic: *CurrentCondition => TargetCondition*

CurrentCondition => always TargetCondition unless NewSituation

An avoid goal refers to a property that is not initially false, but that becomes false, i.e., In terms of temporal logic: *CurrentCondition => not TargetCondition*

An optimize goal refers to a soft property that is to be satisfied.

2.3 Goal Relationships

Goals are related to each other by contribution relationships (partial or complete). Achieving one goal may contribute to achieving another. If the contributing goal is simpler or smaller in scope than the one to which it contributes, the contributing goal is called a sub-goal of the other. The contribution of the sub-goal may be positive or negative (conflict of goals). In the event of a conflict (negative contribution), the achievement of the sub-goal interferes with the achievement of its super-goal. Most popular sub-goal refinements are said to be AND-refinement: if the satisfaction of all of the AND sub-goals is sufficient to ensure the satisfaction of their super-goal. OR-refinement: if satisfaction of any one of the OR sub-goals is sufficient to ensure satisfaction of their super-goal.

3 An Application Case

For managing the Quality Assurance (QA) in its academic programs, Philadelphia University uses a QA agenda, planning its QA management through 16 weeks. The running example used in this study for validating the feasibility of its proposed process is limited to the 7th and 12th weeks because they are the most significant ones. The requirements of the QA agenda are the inputs to the proposed Goal Model generating process: Req-to-GM process.

They are specified using UML use case (Figure 1a) and Swimlane (Figure 1b) diagrams. Using these inputs, an enhanced GM is built up splitting it into behavior goals with their relationships, (2) soft goals with their inherent relationships, and (3) constraints defining relationship between behavior goals and soft goals. This splitting is directed by the syntax shown in Figure 2. The final generated GM is depicted in Figure 3.

4 Target Goal Model

4.1 Enhanced GM by Splitting it into Behavior Goals, Soft Goals, and Constraints

As conclusion to current relevant research works analysis [23], some enhancements to traditional GM are proposed (Figure 2). They are related to GM maintenance levering by splitting it into its three separated aspects: behavior and soft goals and constraints between them. The target GM in Figure 3 is obtained by enhancing GM notations as it is shown in Figure 2.

4.2 GMs Generation Approach – Definitions

The proposed generation approach (Figure 4), is composed of three main components: the inputs that are UML models (Use case description, swimlane diagram), the generation process, which goes through a set of steps for extracting the target GM from requirements, and finally ends with the output, which is the generated GM.

4.3 The Req-To-GM - Process

The Req-to-GM process (Figures 5a, and 5b) takes as inputs requirements specification models (UML use case description and Swimlane Diagram) and generates GM. It is divided into two parallel processes:

- UC-to-GM (Extraction of behavior, soft, and constraints GMs from use case description)
- SL-to-GM (Extraction of soft GM from Swimlane

diagram image).

Each process might do extraction of information, aggregating them into intermediate GMs, and eventually merging the intermediate GMs. In the extraction step, goals (soft, behavior), relationships, quality attributes, and constraints are extracted from the inputs. In the aggregation step, all the outputs from the first phase are aggregated and the behavior, soft, and constraints GMs are built.

Finally, in the merging step, the obtained GMs from the UC-to-GM and SL-to-GM processes are merged into a final output target GM.

4.3.1 Extraction from Use Case Specification. *UC-to-GM*. This process is performed by 6 parallel processes s it is specified in Figure 5c. The process of extracting from the use case document is composed of six parallel sub-processes, which are specified as follows:

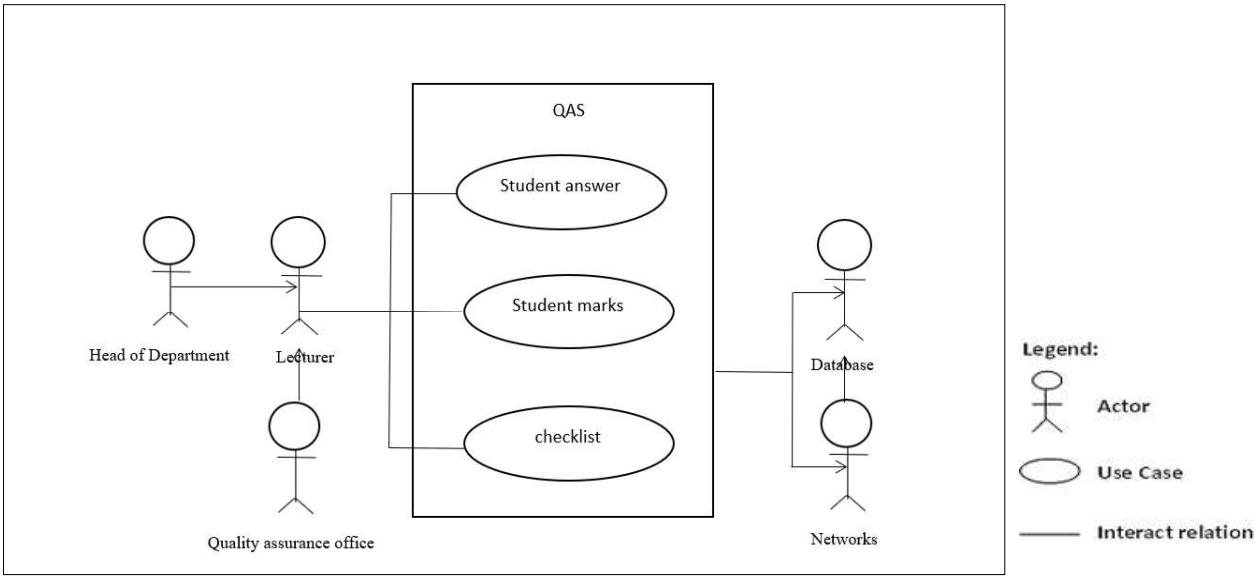


Figure 1a: QA agenda system use case model

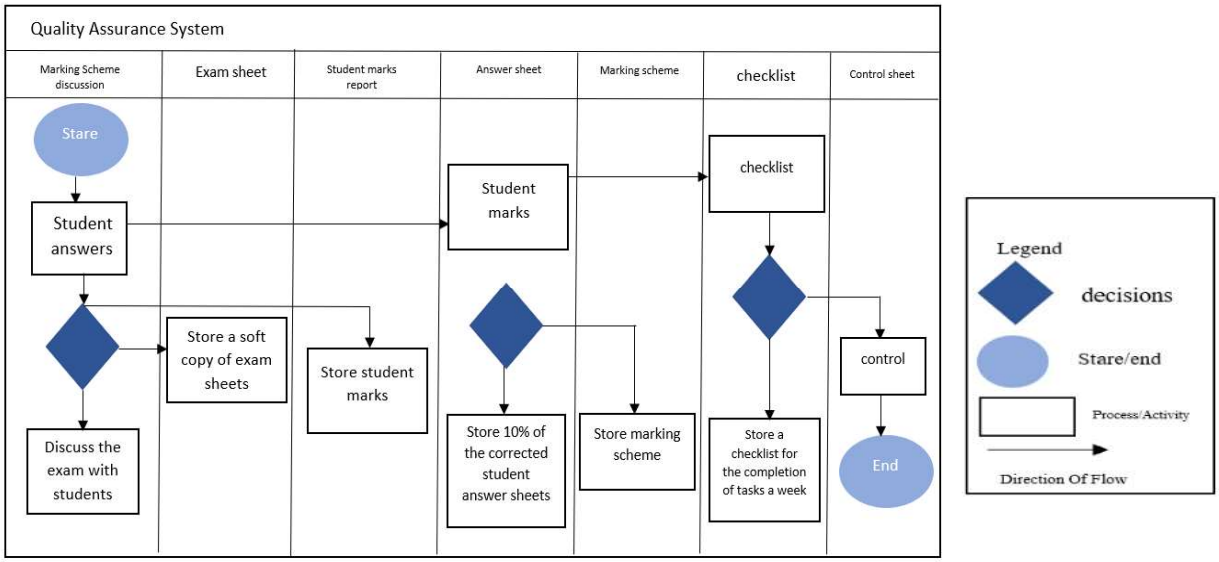
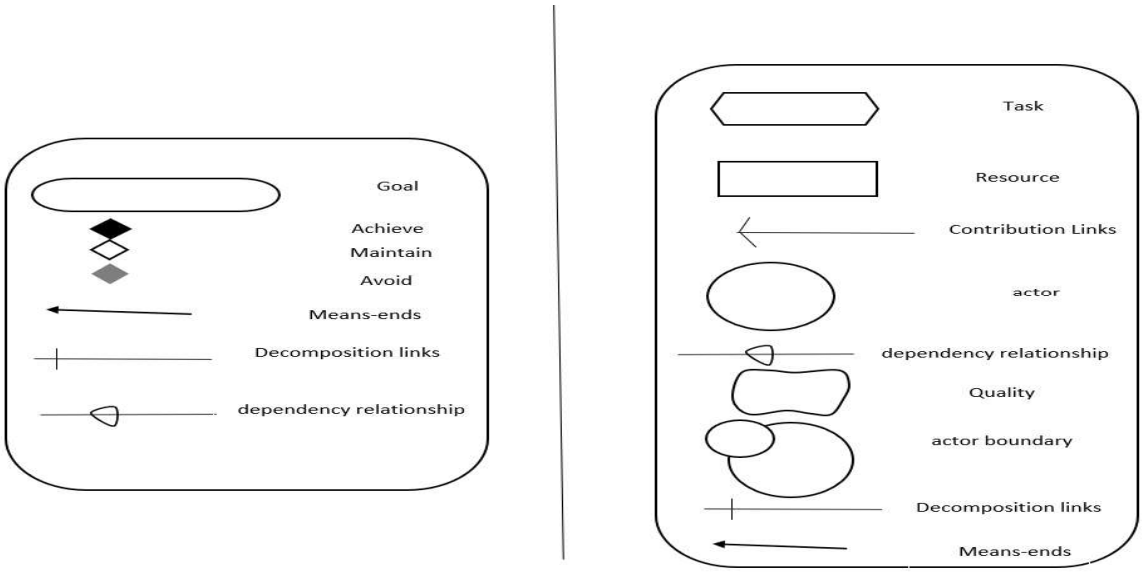


Figure 1b: QA agenda system swimlane diagram



(a) Behavior goals notations

(b) Soft goals notations

Constraints

<list-of- constraints> -- relations between behavior and soft goals

(c) Constraint's notations, defined in a pseudocode language

Figure 2: The enhanced target GM notations

Actor-Extracting: This process extracts actors by reading primary and secondary actor from the use case description (Figure 5d. The actor is represented in the soft goal part.

Actor Boundary Extracting: reading the primary and secondary actor helps identify all parts associated with it in the use case description (Figure 5e). The extracted information is represented in the soft part of the goal model.

Goal Extracting: This process extracts goals behavior from reading the use case name in use case description (Figure 5f). These goals are represented in the behavioral part of the model.

Task Extracting: this process reads the use case description and the task component is obtained, and represented in the soft part of the model (Figure 5g).

Relation (AND / IS-A) Extracting: The use case model contains finite relationships (AND relation / IS-A), specified in the N**ote** attribute that are extracted to represent the relationships between goals and between tasks and also between the actors (Figure 5h).

OR Relation /Goal Type /Quality Extraction: This process extracts *OR Relation /Goal Type /Quality* from note (constraints) notations (Figure 5i).

Constraints Extraction: This process extracts *Constraints* from note (constraints) notations (Figure 5j).

Extraction from swim: lane: This extraction is carried out through *SL-to-GM* process which is performed by single process:

Data Extraction (Figure 5k): This task extracts data from the Swimlane as soft goals.

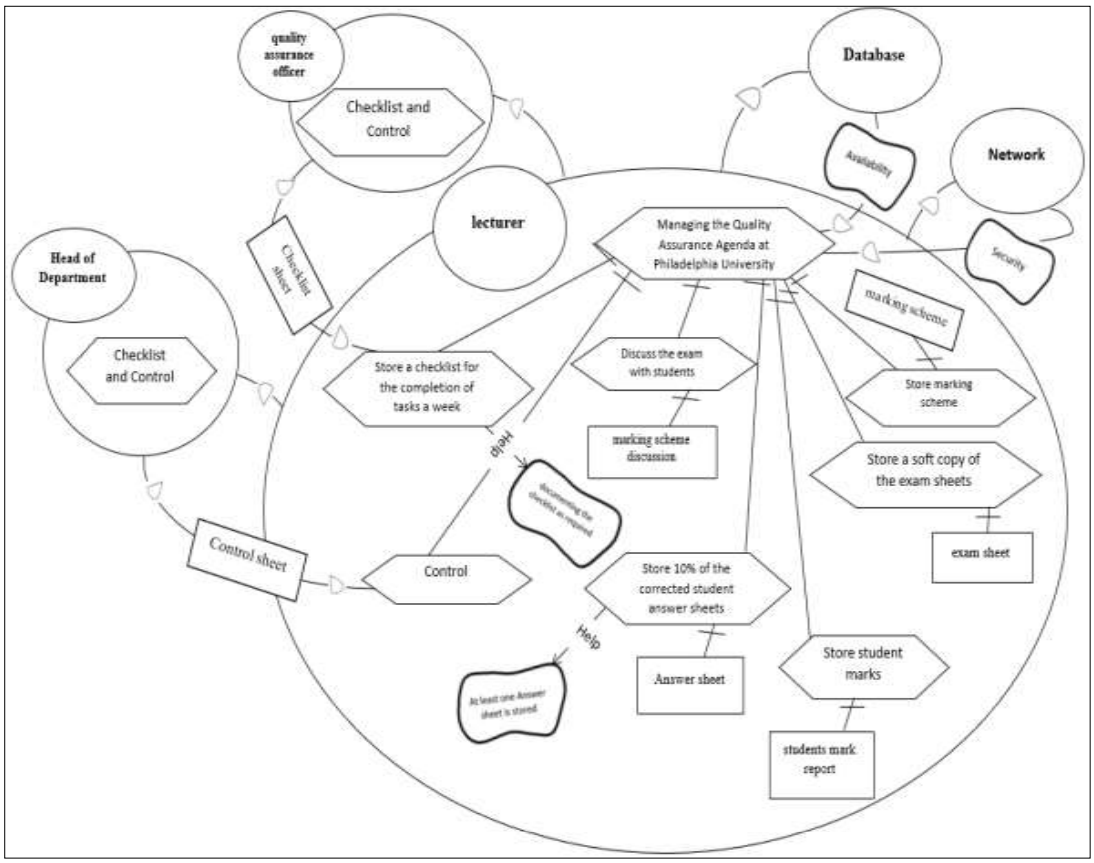
5 Case Study

The feasibility evaluation of the proposed approach was validated by application of the processes Req-to-GM on QA agenda system (paragraph 3). The Figure 6a shows the outputs from the extraction process in UC-to-GM process: actors, actor's boundaries, goals, goals type, relationships between goals, tasks, quality, and constraints. The Figure 6b depicts the soft GM, generated by SL-to-GM process which extracts the data artifacts from the swim lane. The final GM which is the outputs of the Req-to-GM process is depicted in Figure 3.

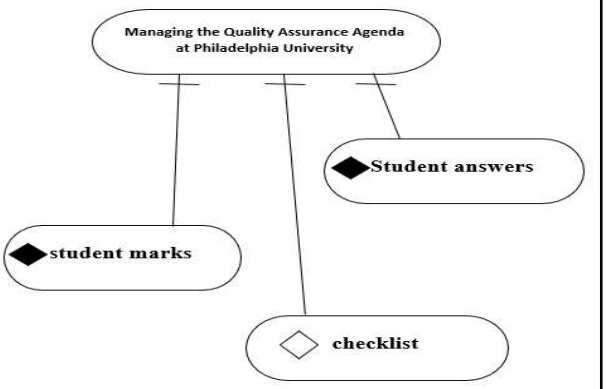
6 Comparison with Similar Works and Evaluation

This section presents a comparison of the proposed approach with some similar works according to the generating process, its inputs and its outputs.

In the work [24], the input is a natural language document which makes it complicated to process as there may be some informal writing leading to lingual mistakes. It requires a further processing to guess the relation between nouns and verbs



(a) Extracted soft GM for QA agenda case



(b) Extracted behavior GM for QA agenda case

1. Student answers ∧ Store marking scheme ∧ Discuss the **exam** with students ∧Store a soft copy of the exam sheets.
2. Student marks ∧ Store 10% of the corrected student answer sheets ∧ Store student marks.
3. Checklist ∧ Store a checklist for the completion of tasks a week ∧ Control.

(c) Extracted constraints GM for QA agenda case

Figure 3: Final generated GM for the case study: QA agenda

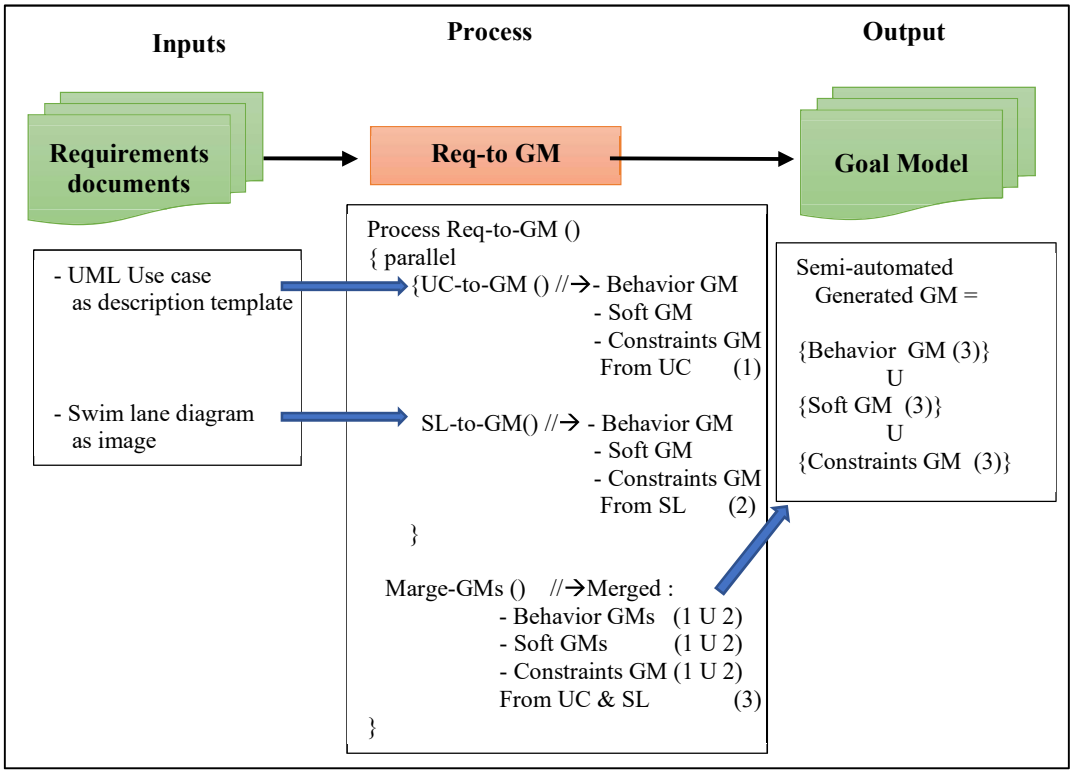


Figure 4: GM generation approach: The Req-to-GM process generates GM from UML use and swim lane diagrams

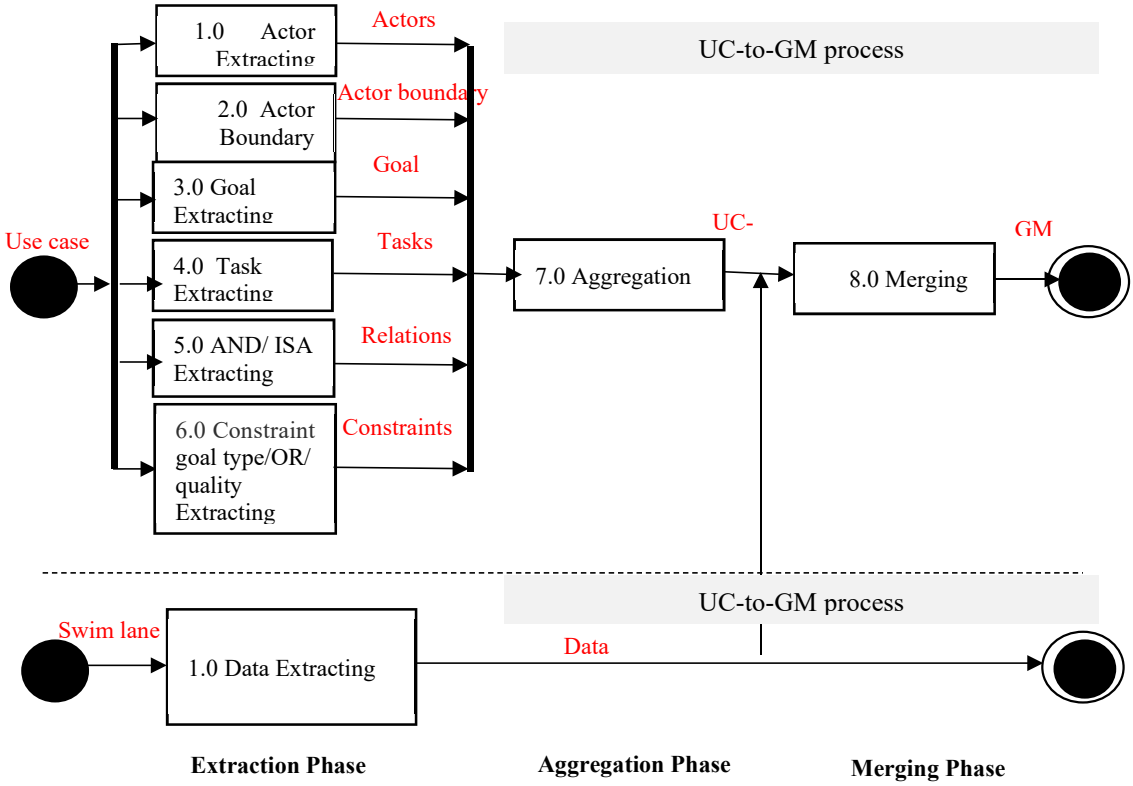


Figure 5a: The Req-to-GM process using UML notations

```
Include goal-model, framework, image //ADT
        UC-to-GM (), Aggregation (), SL-to=GM (), _U_;// functions

Process Req-to-GM (in framework UC-Description, in image SL-image, out goal-model GM)
  goal-model Goal-Model-Elemts, Behavior-GM, Constraints-GM; GMUC, GMSL;
Begin
  Parallel
   {
      ( //UC-to-GM process
        UC-to-GM (UC-Description, Goal-Model-Elemts)
        GMUC ← Aggregation (Goal-Model-Elemts, Behavior-GM) U
                Aggregation (Goal-Model-Elemts, Soft-GM) U
                Aggregation (Goal-Model-Elemts, Constraints-GM))
      )
      (//SL-to-GM process
        SL-to-GM (SL-Image, Goal-Model-Elemts )
        GMSL ←  Aggregation (Goal-Model-Elemts, Soft-GM)
      )
   }// end parallel

   // merging
      GM ← GMUC U GMSL;
  }
End Req-to-GM
```

Figure 5b: The Req-to-GM process using algorithmic notations

```
Include goal, framework, actor, task, relation, constraint //ADT
        Actor-Extracting()  ,Actor-Boundary-Extracting();  Goal-Extracting(),  Task-Extracting(),  Relation-
      and/isa-Extraction(), Constraints-Extraction(); //functions

    Process UC-to-GM (in framework UC-Description, out goal-model Goal-Model-Elemts )
          actor Actors, Actor-Boundary; goal Goals; task Tasks; relation Relations; constraint Constraints;
    Begin
      Parallel
       {
        Actor-Extracting ( UC-Description, Actors)
        Actor-Boundary-Extracting ( UC-Description,  Actor-Boundary)
        Goal-Extracting ( UC-Description,  Goals)
        Task-Extracting ( UC-Description,  Tasks)
        Relation-and/isa-Extraction (UC-Description, Relations)
        Constraints-Extraction (UC-Description, Constraints)
       }
       Goal-Model-Elemts ← {Actors, Actor-Boundary, Goals, Tasks, Relations, Constraints)
    End UC-to-GM
```

Figure 5c: The UC-to-Gm process specification using algorithmic notations

```
Include framework, actor, //ADT
        Read(), Extract() //functions
        Process Actor-Extracting (in framework UC-Description, out actor Actors)
            actor Actors, Primary-actors, Secondary-actors
        Begin
            Read (UC-Description.Actors)
            Actors ← Extract (Primary-actors) U Extract (Secondary-actors)
        End Actor-Extracting
```

Figure 5d: The actor-extracting process specification using algorithmic notations

```
Include framework, actor //ADT
        Read(), Extract-Boundary() //functions
        Process Actor-Boundary-Extracting (in framework UC-Description, out actor Actor-boundary)
        Begin
            Read (UC-Description.Primary-actors, UC-Description.Secondary-actors)
            Actors-boundary ← Extract-Boundary (Primary-actors U Secondary-actors)
        End Actor-Boundary-Extraction
```

Figure 5e: The actor-boundary-extracting process specification using algorithmic notations

```
Include framework, goal //ADT
        Read(), Extract-Goal() //functions
        Process Goal-Extracting (in framework UC-Description, out goal Goals)
        Begin
            Read (UC-Description.Use cases)
            Goals ← Extract-Goal (Use cases)
        End Goal-Extracting
```

Figure 5f: The Goal-Extracting process specification using algorithmic notations

```
Include framework, task //ADT
        Read(), Extract-task() //functions
        Process Task-Extracting (in framework UC-Description, out task Tasks)
        Begin
            Read (UC-Description.Use cases)
            Tasks ← Extract-task (Use cases)
        End Task-Extracting
```

Figure 5g: The task-extracting process specification using algorithmic notations

```
Include framework, relation //ADT
        Read(), Extract-Relation () //functions

    Process Relation-and/Isa-Extraction (in framework UC-Description, out relation
Relations)
    Begin
        Read (UC-Description. Note)
        Relations ← Extract-Relation (relation And) U Extract-Relation (relation Isa)
    End Relation-and/Isa-Extraction
```

Figure 5h: The relation (AND / IS-A)-extracting process specification using algorithmic notations

```
Include framework, attribute //ADT
        Read(), Extract-Attributes () //functions
    Process ORRelation-GoalType-Quality-Extraction (in framework UC-Description, out attribute
                                                    ORTypeQuality)
Begin
    Read (UC-Description. Note)
    ORTypeQuality ← Extract-Attributes (OR, GoalType, Quality)
End ORRelation-GoalType-Quality-Extraction
```

Figure 5i: The ORRelation-GoalType-Quality-Extraction process specification using algorithmic notations

```
Include framework, relation //ADT
        Read(), Extract-Constraints () //functions
    Process Constraints-Extraction (in framework UC-Description, out constraint Constraints)
    Begin
        Read (UC-Description. Note)
        Constraints← Extract-Constraints (Note)
End Constraints-Extraction
```

Figure 5j: The constraints-extraction process specification using algorithmic notations

```
Include image, data //ADT
        Scan(), Data-Pattern-Match () //functions
    Process Data-Extraction (in  image SL-Image, out data Data)
    Begin
        Scan (SL-Image)
        Data ← Data-Pattern-Match (SL-Image)
    End Constraints-Extraction
```

Figure 5k: The Data-Extraction process specification using algorithmic notations

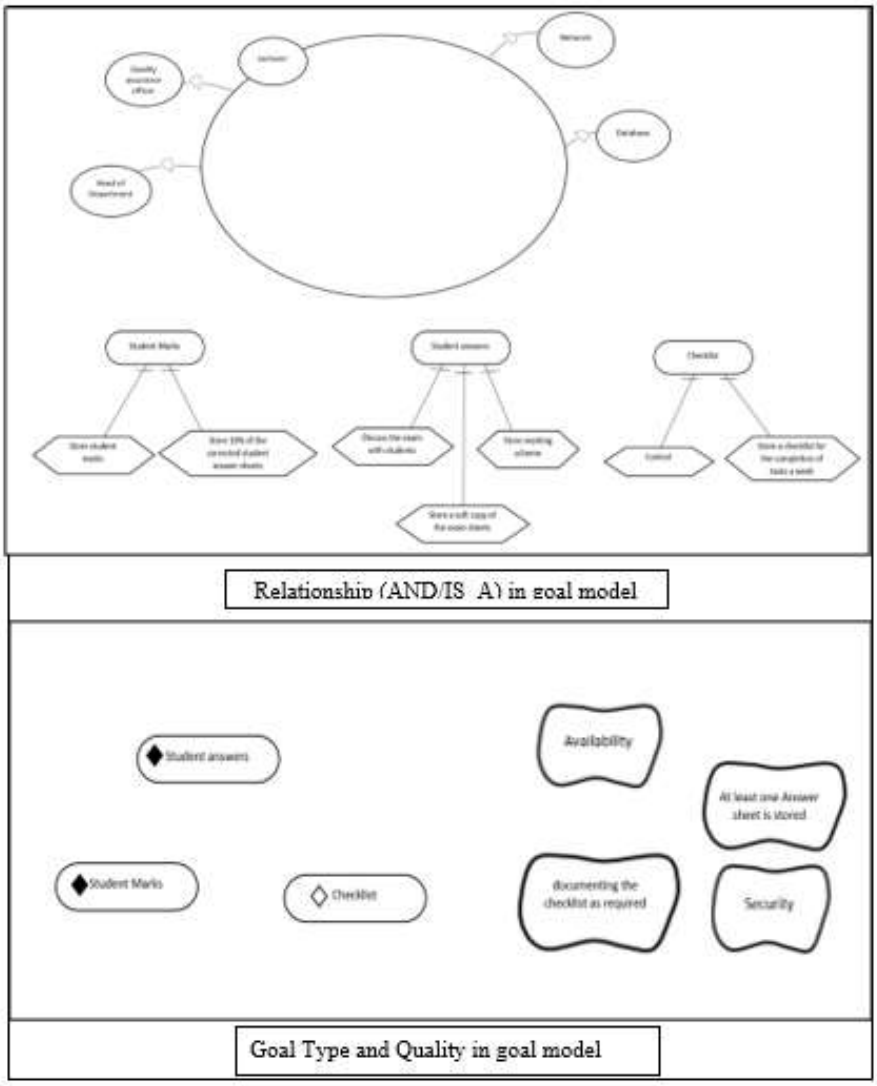


Figure 6a: Generated output by the process UC-to-GM for the QA agenda system

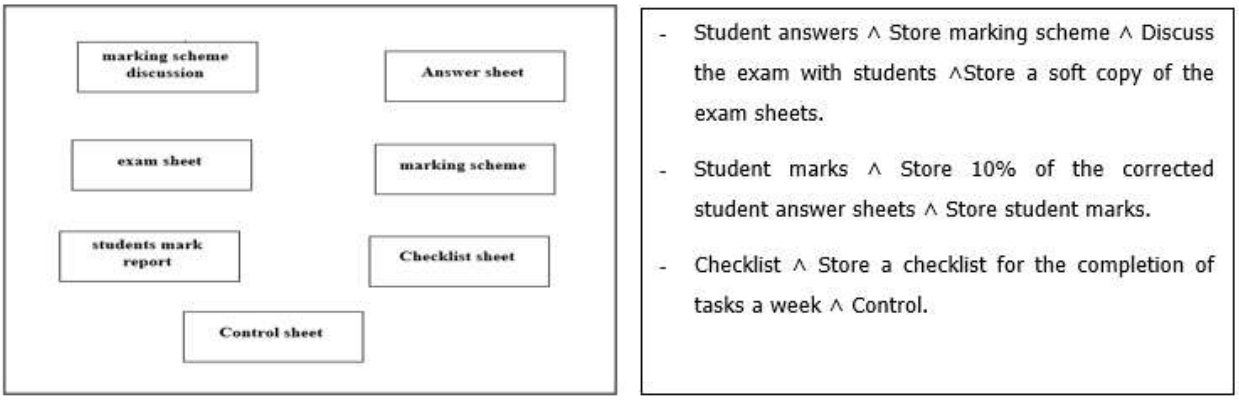


Figure 6b: Generated output by the process SLC-to-GM for the QA agenda system

to finally extract the main goals. Consequently, it takes a considerable time. They do not take into account various types of goals and their automatically generated model was only a single unit.

In [14], the authors used a semi-automatic method that relies on a tool that enables the analyst to ask questions and receive answers, and through these answers, goals and relationships can be determined. However, they have noticed that there is a weakness in this method as it defines the relationships between the objectives as "And relationship". Thus, this method as in the previous one takes time for processing natural language phrases but it's simpler. On the other hand, extracted goals were without types, and the generated models were large single units.

Through [21], again the authors have dealt with the requirements document directly (NLP). Thus, generating a parsing tree through which the analyst can determine the goals then draw the goal model. The intentional tree is often complex and difficult to understand. Finally, there is no distinction between target types.

In [7], once again, entries are written in the natural language. The pipeline technology is used to deal with language keys and the NLP is used to analyze sentences to define goals and relationships. The generated model is based on heuristics which may not generate the optimal model. Their work can't identify the (Or) relation but only the (And).

As a conclusion, the above-mentioned works are NLP-based, they do not take into consideration slicing the generated GMs on behavioral, soft, and constraints aspects leading to difficulty in understanding and maintaining the generated models. This paper proposes a solution to this insufficiency by providing a target cleaned GM separating behavior, soft, and constraints goals, which necessitated addition of some information in the input UML diagrams. The application of the generation process, Req-to-GM, on a real application case revealed its feasibility and effectiveness.

7 Conclusion

The study of similar works revealed that the current methods only generate the basic goal models, which are usually huge, complex containing a lot of goals, tasks and other components. It has been observed that the generation methodologies are not formalized which leads to misunderstanding. Therefore, automated generation methodologies and their input and output remain challenging and need enhancements. This paper proposes the inputs specified semi-formally by UML models. The goal is represented by the use case in first level diagram, and the tasks by the use case in the other levels of the use case diagram. The notes help in representing the rest of the requirements of the components and parts of the goal model (the type of the goal and relationships), it is possible through the swim lanes diagram to represent the data (resources). A formal methodology was developed consisting of two parallel processes, the first process performing extraction from use case description, and the second process performing extraction from swim lane diagram image. The methodology consists of three steps: the extraction, the aggregation, and the merging the proposed methodology feasibility was validated on a real business domain running example. The comparative evaluation with the closest recent works led to clarify the conceptual and practical value of the proposed methodology. This study used UML specification diagrams as input (use case description and swim lane diagram). It could be valuable, in the future, to try other specification models and to compare between the obtained results. The variability and meta modeling of input, process, and output is an important issue.

References

[1] S. Abrahão, E. Insfran, F. González-Ladrón-de-Guevara, M. Fernández-Diego, C. Cano-Genoves, and R. Pereira de Oliveira. "Assessing the Effectiveness of Goal-Oriented Modeling Languages: A Family of Experiments," *Inf. Softw. Technol.*, 116:1-24, 2019.

[2] I. Alexander and L. Beus-Dukic, *Discovering Requirements: How to Specify Products and Services*, John Wiley & Sons, 2009

[3] F. Bozyiğit, Ö. Aktaş, and D. Kılınç, "Linking Software Requirements and Conceptual Models: A Systematic Literature Review," *Eng. Sci. Technol. an Int. J.*, 24(1):71-82, 2021

[4] M. B. Duran and G. Mussbacher, *Reusability in Goal Modeling: A Systematic Literature Review,* Information and Software Techonology, 110:156-173, 2019.

[5] I. A. ElSayed, Z. Ezz, and E. Nasr, "Goal Modeling Techniques in Requirements Engineering: A Systematic Literature Review," *Journal of Computer Science*, Science Publications, 13(9):430-439, 2017.

[6] V. Etemadi, O. Bushehrian, and G. Robles, "Task Assignment to Counter the Effect of Developer Turnover in Software Maintenance: A Knowledge Diffusion Model," *Inf. Softw. Technol.*, 143:106786, 2022.

[7] T. Gunes and F. B. Aydemir. "Automated Goal Model Extraction from User Stories Using NLP", *Proc. IEEE Int. Conf. Requir. Eng.*, 2020:382-387,August 2020.

[8] C. Gupta, P. Inácio, and M. Freire, "Improving Software Maintenance with Improved Bug Triaging," *Journal of King Saud University - Computer and Information Science*s, Part A, 34(10):8757-8764, 2022,

[9] J. Horkoff, N. A. Maiden, and D. Asboth, "Creative Goal Modeling for Innovative Requirements," *Inf. Softw. Technol.*, 106:85-100, 2019.

[10] D. Inkermann, T. Huth, T. Vietor, A. Grewe, C. Knieke, and A. Rausch, "Model-Based Requirement Engineering to Support Development of Complex Systems," *Procedia CIRP*, 84:239-244, 2019.

[11] H. Kaiya, K. Hayashi, and Y. Sato, "Tools for Logging and Analyzing Goal Dependency Modeling," *Procedia Comput. Sci.*, 192:1639-1648, 2021.

[12] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, and F. G. de Oliveira Neto, "Requirements Engineering Challenges and Practices in Large-Scale Agile System Development," *J. Syst. Softw.*, 172:8757-8764**,** 2021.

[13] Y. Lai, A. Gupta, and Y. Zhang, "Goal-Embedded Dual Hierarchical Model for Task-Oriented Dialogue

Generation," *Proceedings of the 23rd Conference on Computational Natural Language Learning*, pp.798-81, 2019
[14] H. Nakagawa, H. Shimada, and T. Tsuchiya, "Interactive Goal Model Construction Based on a Flow of Questions," Transactions on Information and Systems, 103(6):1309-1318, 2020.
[15] A. Pozanco, S. Fern, and D. Borrajo, "Learning-Driven Goal Generation," *AI Communications*, 3(2):137-150, 2018.
[16] S. R. Raheman, H. B. Maringanti, and A. K. Rath, "Aspect Oriented Programs: Issues and Perspective," *J. Electr. Syst. Inf. Technol.*, 5(3):562-575, 2018.
[17] M. R. R. Ramesh and C. S. Reddy, "Metrics for Software Requirements Specification Quality Quantification," *Comput. Electr. Eng.*, 96:107445, 2021.
[18] Z. Ren and K. Dong. "Exploration via Hindsight Goal Generation," 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
[19] M. Sadiq and V. S. Devi. "Fuzzy-Soft Set Approach for Ranking the Functional Requirements of Software," *Expert Syst. Appl.*, 193:11-45, 2021.
[20] V. Santos, H. Mamede, C. Silveira, and L. Reis, "Methodology for Introducing Creativity in Requirements Engineering," *Procedia Comput. Sci.*, 196:27-35, 2022.
[21] H. Shimada, H. Nakagawa, and T. Tsuchiya, "Constructing a Goal Model from Requirements Descriptions Based on Extraction Rules," *Commun. Comput. Inf. Sci.*, 809(no. June):175-188, 2019
[22] L. Yin, Q. Sun, D. Tang, Y. Xu, and L. Shao, "Requirement-Driven Engineering Change Management in Product Design," *Comput. Ind. Eng.*, 168:1-13, 2022.
[23] A. Yousef, S. Ghoul, and M. Taye, "Automatically Generated Goal Model from Requirements: Toward an Enhanced Formalism," Proceedings of ICSIC2022, pp. 1-6, 2022.
[24] C. Zhang Mentor, A. M. Grubb, and M. Chechik, "Toward Automatic Generation of Goal Models using Natural Language Processing," 1:1-6, 2019, [Online], Available: https://www.graphviz.org.

Ahlem Yousef holds a MSc in Software Engineering from Philadelphia University, Amman, Jordan.

She is a member of the research team in the Research Laboratory on Bio-inspired Software Engineering. Her research interest includes: requirements specification, automated specifications generation, and requirements traceability.

Said Ghoul holds a PhD in Software Engineering and a Doctorate es Science in Software Processes.

He is Full Professor of Software Engineering at Philadelphia University, Jordan and Chair of the Research Laboratory on Bio-inspired Software Engineering.

His research interest includes: Bio-inspired Design and Modeling, Variability aspects, and Self-adaptive systems.

Mohammad Mustafa Taye is an Assistant Professor in Software Engineering department at Philadelphia University, Amman, Jordan. He received his Ph.D. degree in Computer Science from De Montfort University, Leicester, UK in 2009, for his dissertation on "Ontology Alignment Mechanisms for Improving Web-based Searching". He also holds MSc. degrees in Computer Science from Amman Arab University for Graduate Study, Amman, Jordan in 2004. Also, holds a bachelor's degree in Computer Science from Irbid National University, Irbid, Jordan, in 2002. Mohammad has been with Philadelphia University since 2009. His research interests include semantic web, ontology, ontologies alignment and matching, ontology languages, artificial intelligence, web service, software requirement, and network.