Goal2UCM: Automatic Generation of Use Case Model from iStar Model

Yilong Yang[1], Younggi Bok[1], Zhuoxi Yang[1], Eric Sheriff[1] a nd Tong Li[2]

[1]*State Key Laboratory of Software Development Environment, School of Software, Beihang University, China*

[2]*Faculty of Information Technology, Beijing University of Technology, China*

Abstract

Goal-oriented modeling is an effective way for modeling and analyzing the requirements of users. It takes stakeholder's intentions as the main clue and analyzes their goals and tasks to construct hierarchical requirements model. Unified Modeling Language (UML) is a de facto standard for system requirements modeling and design. In practice, it is very desirable to have an approach to automatically transform user requirements into system requirements, then automatically generate system prototypes for requirements validation. In ICSE'19 and RE'19, we propose an approach and CASE tool RM2PT, which can automatically generate prototypes from system requirements in UML. In this paper, we focus on filling the gap between user and system requirements. Specifically, we propose an approach Goal2UCM to automatically generate the use case diagram, the system operations and interfaces of use cases from the goal-oriented model iStar based on the model-driven approach. We evaluate the proposed approach with the case study of CoCoME system. Overall, the result is satisfactory. The 93.6% of the iStar model elements can be transformed successfully, and the remaining parts of sub-goals and sub-tasks can be refined and mapped into UML models manually. The proposed approach with the developed CASE tool can be applied to the software industry for requirements engineering.

Keywords

UML, goal model, user requirements, system requirements

1. Introduction

Eliciting and specifying the user requirements is the initial step of requirements engineering. Goal-oriented modeling iStar is a promising way for modeling and analyzing the requirements of users. It provides software developers and designers with a way to model the characteristics of system requirements visually and systematically. In practice, it is very desirable to have an approach to automatically transform user requirements into system requirements, then automatically generate system prototypes for requirements validation. Requirements modeling and automatic prototyping (RM2PT) [1][2][3] is an approach and a CASE tool that can automatically generate a system prototype from UML requirements models. The input models of RM2PT contains a use case diagram, the system sequence diagrams of the use case, the constraints of the system operations in the system sequence diagram, and a conceptual class diagram. The direct transformation from iStar to these UML models is challenging because iStar models lack modeling and describing abilities for the sequences of tasks and conceptual modeling.

Proceedings of the 14th International iStar Workshop, October 18-21, 2021, St. Johns (NL), Canada
EMAIL: yilongyang@buaa.edu.cn (Y. Yang); litong@bjut.edu.cn (T. Li);

Compared with the related works to generate a use case diagram or class diagram from iStar 1.x model, we propose an approach Goal2UCM to automatically generate a use case diagram, the system operations and interface of use cases from iStar 2.0 model based on the model-driven approach. In addition, we implement a iStar 2.0 modeler and Goal2UCM transformer (https://istar.rm2pt.com) , which are integrated with RM2PT to support further prototype generation for requirements validation. We evaluate the proposed approach with the case study of CoCoME system. Overall, the result is satisfactory. The 93.6% elements of iStar models can be transformed successfully, and the remaining parts of sub-goal and sub-tasks can be refined and mapped into UML models manually. The contributions of this paper are summarized as follows:

- We propose an approach Goal2UCM to transform a iStar 2.0 model into a use case model, which includes a use case diagram, the system operations, the message of system operation, and the interface of use cases.
- We implement the proposed approach and a iStar 2.0 modeler as the plugins in our CASE tool RM2PT to support iStar modeling and further prototype generation for requirements validation.
- We demonstrate the effectiveness of the proposed Goal2UCM by CoCoME case study, in which 93.6% of the iStar model elements can be transformed successfully without any extension.

The remainder of the paper is organized as follows: Section 2 presents the related work. Section 3 introduces the iStar and use case models. Section 4 presents the proposed approach of mapping from iStar to use case model. Section 5 demonstrates the proposed approach through CoCoME case study. Section 6 concludes the paper and proposes future work.

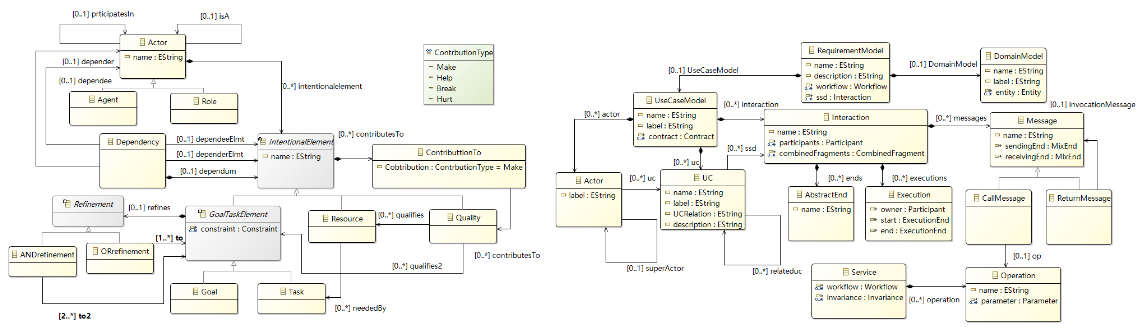
2. Related Work

To the best of our knowledge, the related work is about the transformation from iStar 1.x model to use case diagram or class diagram. The work [4][5] propose a transformation method from iStar to use case diagram, and made a tool JGOOSE to implement the methods. However, they only can the transform iStar 1.x model to a use case diagram, which is not involved other models to describe the specification of the use cases. Another work [6] is to transform a iStar 1.x model to class diagram with OCL constraints. But it does not involve the transformation of use case diagram. The work [7] transform the iStar 1.x model to use case diagram and class diagram. However, all system operations are encapsulated into a same class of the system and the parameters of system operation can not be transformed. The other type of work is about the transformation from natural language to UML models [8][9], it only focuses on the conceptual class diagram but not the other models in UML. In short, the current work can not full support use case model generation. We make a further step to support user requirements validation by transforming the iStar 2.0 model to use case model, which can be used to generate the prototypes to validate the user requirements.

3. iStar and UML Meta-models

The proposed approach is based on the model-driven methodology. Before starting the transformation, the meta-models of iStar and use case model are presented in this section. We use iStar 2.0 (http://istarwiki.org) as the source model and the subset use case model of UML 2.5 (https://www.omg.org/spec/UML) as the target model. The meta-models of iStar and use case model are shown in Figure 1.

iStar Model: The transformation does not cover all elements of the model. For the iStar model, the classifier of the intention subject is *Actor*, which has two sub classes *Agent* and *Role*. Base on the element definitions iStar specification, the *Agent* is the instance of the subject with the *Role*. The *Role* of iStar is semantically equivalent to the *Actor* in UML. Moreover, the *Actor* has several intentions, which can be a *Goal*, *Task*, and *Resource*. The *Goal* is the state that the *Actor* wants to achieve with a clear achievement standard. The *Task* refers to the action that *Actor* wants to implement, usually to achieve a goal. The *Resource* is the physical entity or information entity that the *Actor* needs to execute a task.



(a) iStar Metamodel

(b) UML Metamodel

Figure 1: the subset of iStar and UML Metamodel

UML Model: For the UML model, we do not transform the whole system, but only focus on the use case model, which include the classifiers such as the *Actor* and *UC* in the use case diagram, and the System operation and System service in the system sequence diagram of the use case. The transformation is straightforward, we will map the semantic equivalence classifiers and their relations of iStar to the UML model. We will see transformations rules and algorithm in the next section.

4. Goal2UML: Transformation from iStar to UML models

This section presents how to transform the iStar model into UML model based on the meta-model introduced in the preliminary section. We introduce the transformation rules for the use case diagram and system operations of use case first and then present the transformation algorithm

4.1. Transformation Rules

When transforming the use case diagram, the whole transformation process will be divided into two parts. The first part is the conversion of *Actor*, and the second part is the conversion of *UseCase*. In UML, an *Actor* is an external entity that interacts with the system. It can be a user, an external system that can interact with the system, or a basic device. In the Goal Model, Actors are divided into two categories, *Role* and *Agent*. *Agent* is a specific instance, such as a person, organization, or department. It is not suitable to convert *Agent* to *Actor* in UML because *Agent* is more specific and has more limitations. *Role* is an abstract description of a certain group of people, such as students. It is closer to the meaning of *Actor* in UML, so it can be converted directly.

$$R_1 : \frac{GoalModel.Role(name)}{UML.Actor(name)} \qquad R_2 : \frac{GoalModel.Goal(name)}{UML.UC(name)}$$

The rule R1 describes the process of transforming role in *Goal* model into *actor* in UML.Formula R2 describes the process of transforming *Goal* in Goal Model into *UC* in UML. The *UseCase* describes the function of the system as a series of events and provides valuable observations for the operator. In the Goal Model, a *Goal* is a state that the *Actor* wants to achieve, and there is a clear completion standard. They all describe behavior or state from the perspective of *Actor*, so they can be transformed. However, not all goals can be converted to *UseCase*, only goals at the root can be converted. At present, we do not consider the situation when *Goal* is connected to another *Goal* through Refinement. This problem will be improved in the follow-up work.

The *Task* in iStar represents an action that the *Actor* wants to perform, usually to reach a certain *Goal*. The *Task* is corresponding to the following elements in use case model: *Message* and *Operation*. All of these elements may be involved to complete a single task in the entire *UseCase*. They describe the process of the same task. But if the *Task* is connected by multiple other Intentional Elements with *OrRefinement*, it will not be transformed in our case. The rules R3 and R4 describe the process of transforming *Task* in Goal Model into child elements *CallMessage* and *ReturnMessage* of message in UML.

$$R_3 : \frac{GoalModel.Task(name)}{UML.Message.CallMessage(name)} \qquad R_4 : \frac{GoalModel.Task(name)}{UML.Message.ReturnMessage(name)}$$

The *Task* with the related *Resource* is corresponding to the following elements in UML: *Operation* and *Parameter*. *Task* is transformed into *Operation*, however, if the *Task* is connected by multiple other Intentional Elements with *OrRefinement*, it will not be transformed. The rule R5 describes the process of transforming *Parameter* in Goal Model into *Operation* in UML. The rule R6 describes the process of transforming *Resource* in Goal Model into *Parameter* in UML.

$$R_5 : \frac{GoalModel.Task(name)}{UML.Operation(name)} \qquad R_6 : \frac{GoalModel.Dependency.Resource(name)}{UML.Parameter(name)}$$

4.2. Transformation Algorithm

To use the above transformation rules, we propose the transformation algorithm in this section. It takes an iStar model as the input and outputs UML models. Firstly, it retrieves all the *Role* in iStar model, and then applies the rule R_1 to transform all roles into the actors in UML. For each top goal of the *Role* in iStar, if there is no refinement type connection for the current *Goal*, it indicates that the goal is in the root node section, and the algorithm apply the rule R_2 to transform it into the *UseCase* of UML. If the current *Task* is not connected by other tasks or goals with *OrRefinement*, the rules R_3 and R_4 will be executed, and the current *Task* will be transformed into *CallMessage*, *ReturnMessage* in UML, otherwise, no transformation will be performed. Moreover, the rule R_5 to transform the *Task* in the Goal model into the *Operation* in UML. Then the rule R_6 under the element to transform the *Resource* in the Goal model into a *Parameter* in UML. Finally, the algorithm assembles all the transformed elements of UML and outputs a file *uml.xmi*.

Algorithm 1: Transformation algorithm from iStar to UML models

```
Input  : iStar Model - iStar.xmi
Output : UML Model - UML.xmi
begin
    roles ← retrieve(iStar.xmi);
    for $r \in$ roles do
        apply rule $R_1$;
        elements ← getIntentions(r);
        for $e \in$ elements do
            if e == Goal and isRootGoal(e) then
                apply rule $R_2$;
            end
            if e == Task then
                apply rule $R_3$, $R_4$, $R_5$;
                rs ← getRelatedResource(e);
                for $r \in$ rs do
                    apply rule $R_6$;
                end
            end
        end
    end
    return assemble(UML.xmi);
end
```

5. Evaluation

In this section, we use CoCoME case study (https://www.cocome.org) to evaluate and demonstrate the effectiveness of the proposed approach. We first present a brief introduction about the case study and then show and discuss the transformation results.

5.1. CoCoME Case study

We use the case study CoCoME (Supermarket System) to demonstrate the proposed approach. The main subjects of this system are the cashier and customer. The main intention of cashier is processSale. This part constitutes a simple use case diagram. The goal *processSale* can be expanded to the interaction with customers. First, *makeNewSale* is initiated and then the *enterItem* loop is initiated, requiring the customers to provide the cashier with barcode and quantity information about the goods until it ends. Then we proceed to the next action, *endSale*. Finally, a selection to either *makeCashPayment* or *makeCardPayment* is needed. If cash payment is chosen, a specific amount is required. On the other hand, if card payment is chosen, the card account number, expiry date and fee is required. Due to space constraints, this chapter can not give a good introduction to this example. Therefore, we have made relevant web pages with detailed instructions (http://istar.rm2pt.com).

5.2. Evaluation Results and Discussion

The transformation results are divided into the following two parts. The first part is a use case diagram, which describes the relationship between user *Cashier* and his/her *UseCases*. Then the system operations of the use cases *processSale*. The transformation success rate from the Goal model to the top-level use case diagram can be reaches 100%. The success rate of the system operation and message can be reaches 87.2%.

Table 1
Transformation result table from Goal model to Use Case diagram

Name	Actor	Use case	Use case diagram relationships	Success rate
Goal2UCM	2/2	4/4	4/4	100%

* N / M N: represents the number of elements that can be converted by the algorithm, M: represents the example of cope. There are several elements in this part

Use Case diagram: transform role in Goal model into Actor in UML. The goal in the root node part of the Goal model is transformed into UML UC , that is, the goal without Refinement type connection is transformed into UC. And connect them to their respective actors.

Table 2
Conversion results from Goal model to system sequence diagram

Name	System operation	System service	Success rate
Goal2UCM	35/47	17/17	87.2%

System sequence diagram: each task in the Goal model is transformed into the corresponding interaction (Call Message, Return Message, Execution) and Service (Operation) in UML, and the Resource of Dependency in the Goal model is transformed into the corresponding Service (Operation) in UML.

Table 3
Transformation results from target model to UML model

Name	Use Case diagram	System Operations	Success rate
Goal2UCM	100%	87.2%	93.6%

Table 3 describes the overall transformation success rate from the Target model to Use Case model. The transformation success rate of the top-level use case diagram is 100%. Due to the lack of the information about the parameters, the transformation success rate of system operations is 87.2%, and the overall transformation success rate is 93.6%.

6. Conclusion and Future Work

In this paper, we propose an approach Goal2UCM for transforming the iStar to use case model in UML, which contains a use case diagram and the system operations with its parameters of the use cases. Overall, the result is satisfactory. The 93.6% of iStar model elements can be transformed successfully. The proposed approach is implemented and integrated with RM2PT to support requirements validation. However, the results show that some elements of UML can not be directly generated due to the lack of the information in the iStar model. In the future, we will try to extend iStar model and transformation rules with the sequence marks of the tasks to support system sequence diagram models generation as well as the contract of system operation and conceptual model to fully support the prototype generation.

References

[1] Y. Yang, X. Li, W. Ke, Z. Liu, Automated prototype generation from formal requirements model, IEEE Trans. Reliab. 69 (2020) 632–656.

[2] Y. Yang, X. Li, Z. Liu, W. Ke, RM2PT: a tool for automated prototype generation from requirements model, in: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, ICSE 2019, May 25-31, 2019, pp. 59–62.

[3] Y. Yang, W. Ke, X. Li, RM2PT: requirements validation through automatic prototyping, in: 27th IEEE International Requirements Engineering Conference, IEEE, 2019, pp. 484–485.

[4] V. F. A. Santander, J. Castro, Deriving use cases from organizational modeling, in: 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002), 9-13 September 2002, IEEE Computer Society, 2002, pp. 32–42.

[5] G. C. L. Geraldino, V. F. A. Santander, The JGOOSE tool, in: J. Pimentel, J. P. Carvallo, L. López (Eds.), Proceedings of the 12th International i* Workshop co-located with 38th International Conference on Conceptual Modeling (ER 2019), CEUR-WS.org, 2019.

[6] J. Castro, J. Mylopoulos, F. M. R. Alencar, G. A. C. Filho, Integrating organizational requirements and object oriented modeling, in: 5th IEEE International Symposium on Requirements Engineering (RE 2001), IEEE Computer Society, 2001, pp. 146–153.

[7] J. F. Castro, F. M. R. Alencar, V. F. A. Santander, C. T. L. Silva, Integration of i* and Object-Oriented Models, in: Social Modeling for Requirements Engineering, The MIT Press, 2010, pp. 457–484.

[8] C. Arora, M. Sabetzadeh, L. C. Briand, F. Zimmer, Extracting domain models from natural-language requirements: approach and industrial evaluation, in: Proceedings of the ACM/IEEE 19th, ACM, 2016, pp. 250–260.

[9] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, S. Brinkkemper, Extracting conceptual models from user stories with visual narrator, Requir. Eng. 22 (2017) 339–358.