PAPER *Special Section on Knowledge-Based Software Engineering*

Interactive Goal Model Construction Based on a Flow of Questions

Hiroyuki NAKAGAWA[†a)], ***Member***, **Hironori SHIMADA**[†], ***Nonmember***, ***and*** **Tatsuhiro TSUCHIYA**[†], ***Member***

SUMMARY Goal modeling is a method that describes requirements structurally. Goal modeling mainly consists of two tasks: extraction of goals and organization of the extracted goals. Generally, the process of the goal modeling requires intensive manual intervention and higher modeling skills than the process of the usual requirements description. In order to mitigate this problem, we propose a method that provides systematic supports for constructing goal models. In the method, the requirement analyst answers questions and a goal model is semi-automatically constructed based on the answers made. We develop a prototype tool that implements the proposed method and apply it to two systems. The results demonstrate the feasibility of the method.

key words: *requirements analysis, goal models, interactive tools*

1. Introduction

Requirements analysis contains requirements extraction, in which an analyst elicits requirements of a system under development. Since the requirements of a software system are often vast and diverse, it is usually difficult to extract all of the requirements and construct a model that relates the extracted requirements to each other. To support the requirements extraction, several requirements models are proposed. For example, use case diagrams and goal models, such as KAOS [1], i* [2], NFR [3], and AGORA [4], can be used to visualize requirements in a structural manner. While the former is mainly used for addressing functions that a system should provide, the latter is used for visualizing relationships between requirements. In this paper we focus on the latter, i.e., the goal model.

A goal model describes a set of goals and relationships among the goals. The goal model can visualize the structure of requirements; however, constructing a goal model is still difficult, especially in the following two aspects:

(1) **Constructing a correct goal model**: Goals are decomposed into smaller goals in a goal model. Goal relationships have to be defined among these goals so that the relationships can correctly reflect requirements. However, determining such correct relationships is not an easy task.
(2) **Extracting requirements without omission**: All requirements should be extracted in the requirements analysis phase. If the analyst fails to extract an important requirement, the omission might lead to a serious problem in later development stages. However, requirements are not always explicit and thus are often subject to omission.

The objective of a series of our studies is to establish a systematic goal model construction process. In our previous work, we proposed a method of extracting goals based on extraction rules to accomplish the second aspect [5]. The extraction rules allow requirements analysts to identify some implicit goals in requirements documents. In this paper, we mainly focus on the first aspect. We propose a method that supports goal extraction from requirements documents and goal model construction. In our method, an analyst answers to a flow of questions. Our method constructs a goal model based on the answers. Since the questions are asked interactively, the goal model is constructed semi-automatically. We develop a prototype tool that implements the proposed method. We evaluate our method by applying the method to two examples of systems using the tool.

The contribution of this paper is thus the proposal and evaluation of the goal model construction method. The semi-automated construction provided by the method alleviates many of the problems with manual construction. Manual construction is inherently subject to human errors, which may result in inaccurate or incomplete models. The automatic nature of the proposed method can be useful for reducing such human errors. It also reduces the fluctuation in quality of resulting models caused by difference in skills of analysts.

The remaining part of this paper is structured as follows: Sect. 2 explains the background of our research; Sect. 3 describes our method for constructing goal models semi-automatically; Sect. 4 reports experimental results of goal model construction; Sect. 5 discusses our approach based on the experimental results; Sect. 6 describes potential threats to validity of the results; and finally, Sect. 7 concludes this paper and describes future work.

2. Background

Some requirements models introduced in previous studies, such as the use case diagram and the goal model, can be used for requirements visualization. A goal model contains goals, which represent requirements and are structured in a directed acyclic graph (DAG). In the KAOS model, for

Manuscript received August 19, 2019.
Manuscript revised January 27, 2020.
Manuscript publicized March 6, 2020.

[†]The authors are with Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: nakagawa@ist.osaka-u.ac.jp

DOI: 10.1587/transinf.2019KBP0015

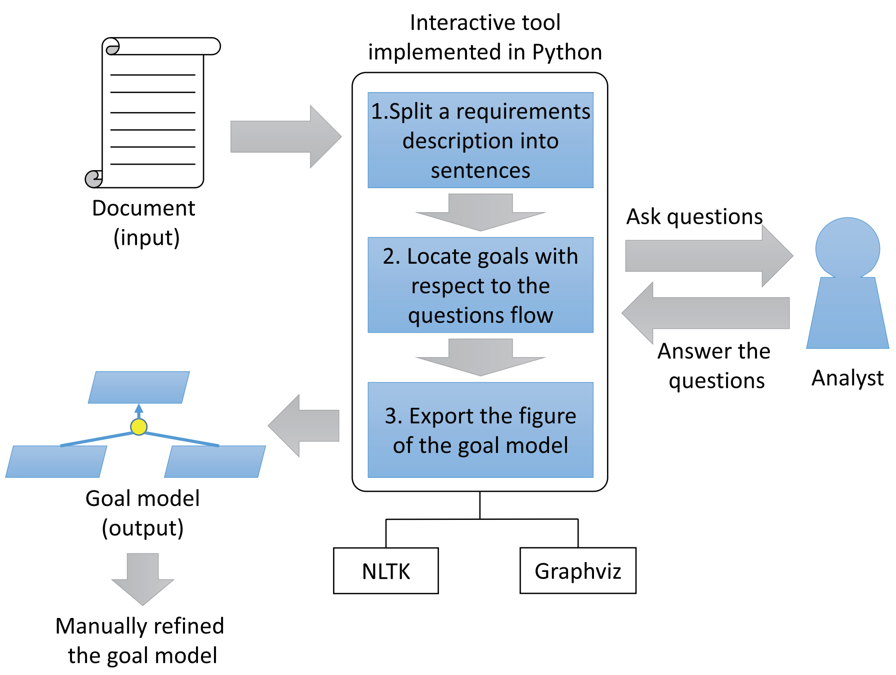

Interactive tool implemented in Python
Document (input)
1.Split a requirements description into sentences
2. Locate goals with respect to the questions flow
3. Export the figure of the goal model
Ask questions
Answer the questions
Analyst
Goal model (output)
NLTK
Graphviz
Manually refined the goal model

Fig. 1 Semi-automated goal model construction process.

example, the goal located at the root node represents the purpose of the system development and is in turn refined into subgoals represented by child nodes of the root. The subgoals are also refined into smaller subgoals.

Several studies address the tasks of requirements extraction and goal model construction. Pimentel et al. [6] developed a web tool that supports goal modeling and statechart derivation from goal models. Rahimi et al. [7] proposed the process of automatic extraction and visualization of quality concerns. Our method extracts not only quality concerns, but also functional requirements including exception handling. Nguyen et al. [8] proposed a semi-automated goal-use case model extraction method which is based on extraction rules. In our approach, we focus on a goal model and its construction based on a questions flow. Franch et al. [9] proposed a systematic method of constructing *i** strategic dependency models based on rules, criteria, questions, and patterns. Since the method is tightly tailored to *i** strategic dependency models, it is not possible to directly apply it to goal model construction.

Several studies address analysis or evaluation of goal models. Zee et al. [10] provided a framework for tracing elements of goal models described in the goal-oriented requirements language (GRL) to rationalize the models. Yu et al. [11] proposed a method of mapping goal models to feature models based on mapping rules. They also proposed an algorithm for mapping dependencies among actors to the feature model. Santos et al. [12] proposed an alternative concrete syntax for KAOS goal models and claimed that the syntax improves cognitive effectiveness and increases semantic transparency. Liaskos et al. [13] assessed user perception of the meaning of the contribution link construct of a goal modeling language. Han et al. [14] proposed the adapt-requirement model, which integrates goal models and problem frames. They also provided UML profile to represent the adapt-requirement model, facilitating use and management of the adapt-requirement model. Piras et al. [15] proposed the Agon framework that enables systematic acceptance requirements analysis. Nguyen et al. [16] proposed a method of adapting a constrained goal model (CGM) tool requirements changes. While most of the studies focus on analyzing or reusing goal models, our study focuses on supporting goal model construction.

3. Semi-Automated Goal Model Construction

3.1 Overview

This section presents an interactive goal model construction method that allows an analyst to generate a goal model semi-automatically by answering a questions flow. Figure 1 illustrates an overview of our goal model construction process. The input of the process is a document written in a natural language that contains requirements of a system; the output is a constructed goal model. We define a set of questions and implement a tool that interactively asks one of questions to an analyst. The tool is implemented in Python and uses NLTK [17] for the natural language processing. The tool first splits a document into sentences. Each sentence may contain requirements that correspond to a goal. The tool then asks questions based on the questions flow to the analyst. By using the answers obtained from the analyst, the tool decides the location of goals in a goal model. The tool visualizes the constructed goal model using Graphviz [18], which is an open source tool for drawing graphs specified in the DOT language.

Table 1 A set of questions defined in this study.

Question	Label
What is the purpose of the system?	askPurpose
What is the goal description of the requirement?	isGoal
What is the type of the goal?	selectType
Is the goal X a subgoal of the goal Y?	isSubgoal
Can these goals be aggregated?	canAggregateGoals
What is the parent goal description of the child goals?	askParentGoal

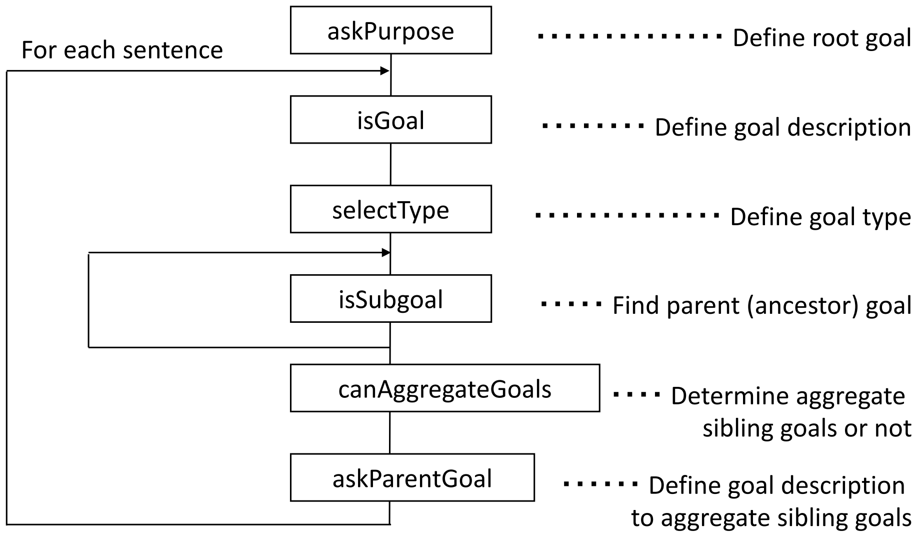


Fig. 2 Each question and its role in questions flow

The generated goal model may require further modifications due to some reasons, such as the incompleteness of the input document. In this case, the analyst manually adds or refines the generated goal model.

3.2 Questions

In order to construct a goal model, we have to define goals and find relationships among the goals. Before defining a set of questions, we set the following assumptions in our current study: (1) the constructed goal model has only AND-refinement links; (2) each sentence in the input document contains at most one goal. Actually, these two assumptions are not strong because we can replace some AND-refinement links to OR-refinement links and can decompose the goal corresponding to a sentence after generating the goal model. The aim of defining two assumptions is to simplify the evaluation of the feasibility of our approach.

Base on the assumptions, we define six questions to construct a goal model. Table 1 lists these questions, while Fig. 2 shows their role in the questions flow. The first three questions, i.e., askPurpose, isGoal, and selectType, are used to define goals from the input requirements description. The other questions i.e., isSubgoal, canAggregateGoals, and askParentGoal, are used to find relationships among goals. The details of the questions are as follows:

- **askPurpose (What is the purpose of the system?):** The analyst answers the purpose of the system in a short description. This question is intended to define the root goal of a goal model. For example, if we develop an ATM system, a possible answer to this question is "Banking transaction automation."
- **isGoal (What is the goal description of the requirement?):** This question asks the goal description for a sentence taken from the requirements description. As stated, the proposed method regards each sentence of the input requirements description as a candidate for a goal. For the case of the ATM system, the sentence "A customer must be able to make a cash withdrawal from any suitable account linked to the card" can be summarized as goal description "Make cash withdrawal." If the sentence does not represent requirements, the analyst skips the question to proceed to the next sentence.
- **selectType (What is the type of the goal?):** The question asks the type of an extracted goal. By doing so, each goal is classified into one of the three types: *normal behavior*, *exception handling*, and *NFR (Non-Functional Requirements)*. In requirements analysis, exceptional case consideration is lacked because the analyst tends to believe that the system and its environment behave as expected [19]. Therefore, extracted requirements in exceptional case (written in a requirements description) should be specified in a goal model to remind the existence of non-extracted requirements in an exceptional case. NFR also should be extracted to ensure the quality of the system.

Algorithm 1 Overview of the question flow

```
// define the root goal
CREATE(root)
root.description ← askPurpose.answer
for each sentence in a requirements description do
    if isGoal.answer then
        CREATE(ngoal) //define a new goal
        ngoal.description ← isGoal.answer
        // set the type of the goal
        ngoal.type ← selectType.answer
        switch ngoal.type do
            case NB (=normal behavior type)
                LOCATENB(ngoal)
            case EH (=exception handling type)
                LOCATEEH(ngoal)
            case NFR
                LOCATENFR(ngoal)
```

- **isSubgoal (Is the goal *X* a subgoal of the goal *Y*?):** The analyst answers this question by "Yes" or "No". If the answer is "Yes," the goal *X* is located as a descendant goal of the goal *Y*. If the answer is "No," the question is repeated until the parent goal is found.
- **canAggregateGoals (Can these goals be aggregated?):** The question specifies a set of sibling goals and asks whether these sibling goals need to be aggregated. In the ATM example, sibling goals "Make cash withdrawal" and "Make deposit", which represent the same transaction, can be aggregated. This question enables to aggregate such goals.
- **askParentGoal (What is the parent goal description of the child goals?):** The question concerns sibling goals that are aggregated and is skipped if they are not aggregated. The analyst answers the question with a parent goal description. For the above ATM example, the answer could be "Perform transaction." Answering this question yields a new goal and adds it to the goal model as a parent of the sibling goals.

3.3 Questions Flow

We present the questions flow and the goal model construction based on it in the form of pseudo-code (Algorithms 1 and 2). Algorithms 1 represents the overview of the question flow. The analyst first answers the askPurpose question ("What is the purpose of the system?"). This answer is located as the root goal in the goal model. The tool then asks questions for each sentence in the input document. If a sentence contains a requirement, i.e., a goal, the goal is created and inserted in an appropriate position in the goal model. In this case, the tool first creates a goal whose description is the answer of the isGoal question ("What is the goal description of the following requirement?"). Next, the tool finds the goal type of the goal by asking the selectType question. As described in Sect. 3.2, we categorize goals into three types: normal behavior (NB), exception handling (EH), and non-functional requirements (NFR). Algorithm 2 illustrates the process for locating a normal behavior goal. Other two types of goals are handled by algorithms similar to Algorithm 2, implemented as the functions named "LocateEH(*ngoal*)" and "LocateNFR(*ngoal*)", respectively. These functions have the same structure of "LocateNB(*ngoal*)" (Algorithm 2) with the difference of the goal type to be gathered.

Algorithm 2 LOCATENB(*ngoal*): locating a normal behavior goal

```
var tmp ← the most recent defined goal prior to ngoal
// ask whether ngoal should be a subgoal of tmp or not
if isSubgoal(ngoal, tmp).answer then
    // locate ngoal as the subgoal of tmp
    tmp.addSubGoal(ngoal)
else
    while True do
        if tmp is the root goal then
            FINDPARENT(ngoal, tmp)
            break
        else
            tmp ← tmp.getParentGoal()
            if isSubgoal(ngoal, tmp).answer then
                // ask whether subgoals of tmp can be aggregated
                subs ← tmp.subGoals
                if canAggregateGoals(subs).answer then
                    // insert a new goal to aggregate subgoals
                    CREATE(insGoal)
                    insGoal.description
                        ← askParentGoal(tmp.subGoals).answer
                    insGoal.setSubGoals(tmp.subGoals)
                    tmp.releaseSubGoals(tmp.subGoals)
                    tmp.addSubGoal(ngoal)
                else
                    FINDPARENT(ngoal, tmp)
                break
```

To determine the suitable position of goals, Algorithm 2 first traces upward in the goal model and then moves downward to focus to the most suitable position. The "findParent(*g1, g2*)" function in Algorithm 2 has the role of executing the latter part. The function searches for the most suitable position, that is, the function finds the most suitable parent goal to locating the goal g1 by continuous asking the "isSubgoal" question. The question starts from the inquiry of the relationship between *g1* and *g2*, i.e., whether the goal *g2* is the parent goal of *g1* or not. This question is repeated by changing the candidate parent goal. The candidate parent goal is changed by following the breadth-first search in the subtree whose root note is *g2*. The question stops when all of the same depth goals are not the parent goal of *g1*. Then the goal *g1* is located to the suitable position according to the inquiry results.

As a running example, let us consider part of a requirements description of an ATM system [20] as shown below.

(1) The ATM must be able to provide the following services to the customer: (2) 1. A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $20.00. (3) Approval must be obtained from the bank before cash is dispensed. (4) 2. A customer must be able to make a deposit to any account

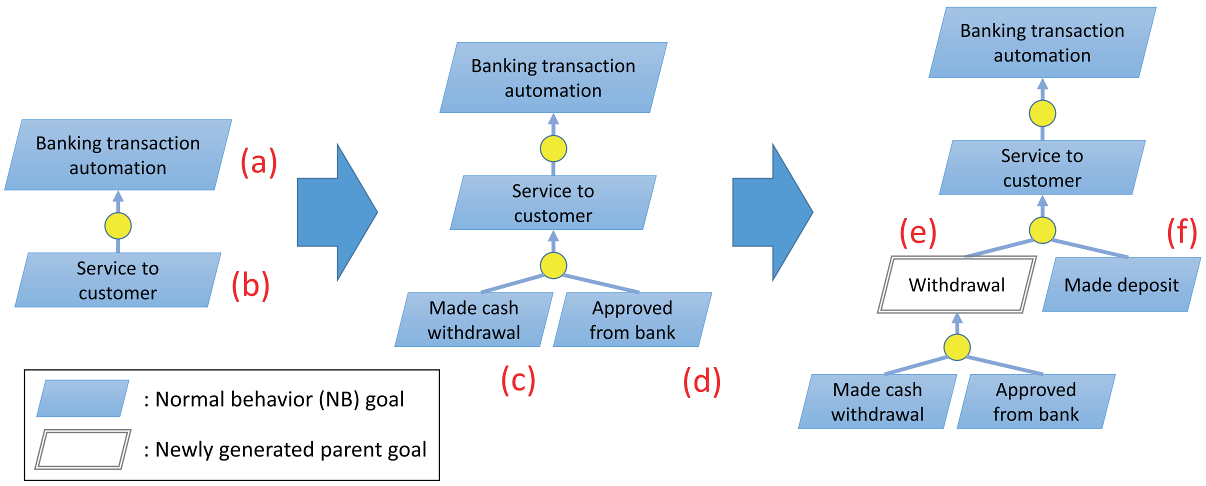


Fig. 3 An example of a goal model construction.

linked to the card, consisting of cash and/or checks in an envelope.

The description contains four sentences. We first answer the *askPurpose* question so that the root goal "Banking transaction automation" ((a) in Fig. 3) is created. The goal "Service to customer" ((b) in Fig. 3) corresponding to the first sentence is created by asking the next question. Since only the root goal is defined in the goal model at this time, the goal (b) is defined as a subgoal of the root goal. The goal "Make cash withdrawal" ((c) in Fig. 3) is then derived from the second sentence. The isSubgoal question in Algorithm 2 is used to decide the location of the goal. In this example, the goal is determined as a subgoal of (b); thus, it is attached to (b). Next, the goal "Approved from bank" ((d) in Fig. 3) is created from the third sentence and added to the model as a subgoal of (b). After that, the goal "Make deposit" ((f) in Fig. 3) is derived from the forth sentence. The goal is added to the model as a child node of goal (b). Finally, (c) and (d) are aggregated, resulting in creating their new parent goal "Withdrawal" ((e) in Fig. 3).

4. Experiment

To evaluate our goal model construction process, we constructed goal models for two sample software applications, an ATM system [20] and a meeting scheduling system [21]. We used two documents provided in [20] and [21]. This experiment consisted of two parts. First, we constructed goal models with two methods, i.e., manual construction and semi-automatic construction using our process, and then compared the two models (Exp 1). In the semi-automatic goal model construction, we used the tool that implements the proposed process. Next, we quantitatively compared goal model constructions of several examinees. The examinees constructed goal models manually and semi-automatically (Exp 2). This section reports the results of these experiments.

4.1 Exp 1

Comparing the two goal models for each of the systems, we found that the relationships among goals are correctly defined. Two goal models constructed for the ATM system are displayed in Fig. 4. Hereafter we call the model constructed using the proposed process the "Auto" model and the model constructed manually the "Manual" model. Although Regions (a) to (e), surrounded by red dotted lines, are structurally different between the two models, careful observation reveals that all of them represent almost the same relationships.

In regions (a) and (d), some goals appear only in the Auto model. For example, while a goal in Region (d) of the Auto model aggregates some goals about ATM services, this goal does not appear in the Manual model. The difference in goal decomposition granularity can be seen in Regions (b), (c) and (e). One of the goal model construction guidelines [21] addresses that goals should be decomposed until all of the subgoals can be achieved by individual single actors including the system to be developed. The proposed process, however, is applied under the assumption that at most one goal is derived from each sentence of the input document. This assumption causes the differences of the goal granularity.

Difference of parent-child relationships is observed in (f). A goal in Region (f) of the Manual model is placed in different regions in the Auto model, i.e., Regions (f-1) and (f-2). The separation is caused by the description of the input document. These separated goals are described in different paragraphs of the document. In our algorithm, the isSubgoal question, which asks whether a specified existing goal is the parent goal of a new goal, is continuously asked from the most recently added goal. Thus, when more than one goal is appropriate as a parent goal of the new goal, the new goal will be attached to one of the candidate goals that is visited first.

Table 2 lists the quantitative evaluation results. As de-

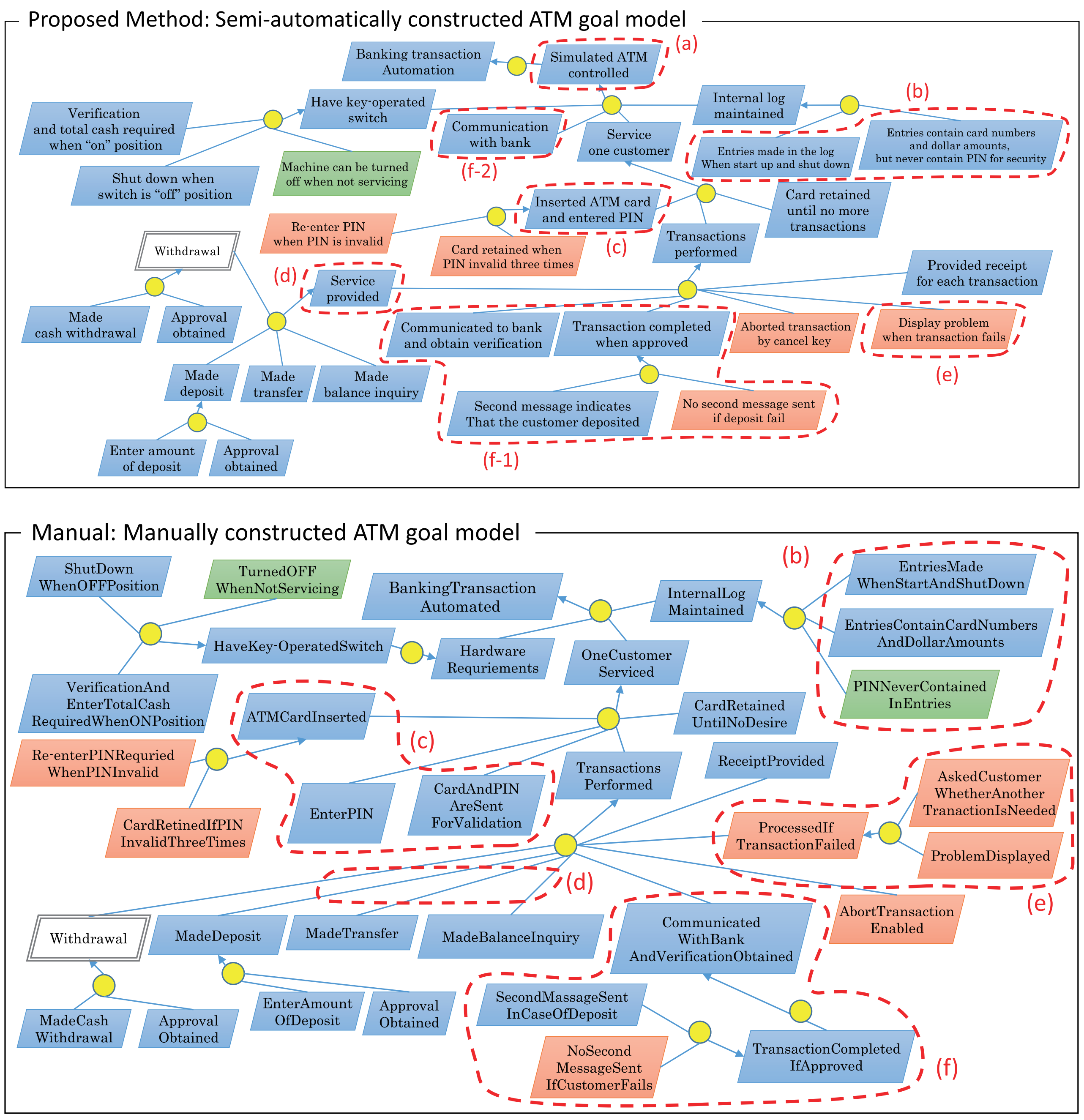

Proposed Method: Semi-automatically constructed ATM goal model
Banking transaction Automation
Simulated ATM controlled
(a)
Verification and total cash required when "on" position
Have key-operated switch
Communication with bank
Service one customer
Internal log maintained
(b)
Entries made in the log When start up and shut down
Entries contain card numbers and dollar amounts, but never contain PIN for security
Shut down when switch is "off" position
Machine can be turned off when not servicing
(f-2)
Inserted ATM card and entered PIN
Card retained until no more transactions
Re-enter PIN when PIN is invalid
Card retained when PIN invalid three times
(c)
Transactions performed
Withdrawal
(d)
Service provided
Provided receipt for each transaction
Made cash withdrawal
Approval obtained
Communicated to bank and obtain verification
Transaction completed when approved
Aborted transaction by cancel key
Display problem when transaction fails
(e)
Made deposit
Made transfer
Made balance inquiry
Second message indicates That the customer deposited
No second message sent if deposit fail
Enter amount of deposit
Approval obtained
(f-1)
Manual: Manually constructed ATM goal model
ShutDown WhenOFFPosition
TurnedOFF WhenNotServicing
BankingTransaction Automated
InternalLog Maintained
(b)
EntriesMade WhenStartAndShutDown
EntriesContainCardNumbers AndDollarAmounts
PINNeverContained InEntries
HaveKey-OperatedSwitch
Hardware Requriements
OneCustomer Serviced
VerificationAnd EnterTotalCash RequiredWhenONPosition
ATMCardInserted
CardRetained UntilNoDesire
(c)
Re-enterPINRequried WhenPINInvalid
EnterPIN
CardAndPIN AreSent ForValidation
Transactions Performed
ReceiptProvided
CardRetinedIfPIN InvalidThreeTimes
AskedCustomer WhetherAnother TranactionIsNeeded
ProcessedIf TransactionFailed
ProblemDisplayed
(d)
(e)
Withdrawal
MadeDeposit
MadeTransfer
MadeBalanceInquiry
Communicated WithBank AndVerificationObtained
AbortTransaction Enabled
MadeCash Withdrawal
Approval Obtained
EnterAmount OfDeposit
Approval Obtained
SecondMassageSent InCaseOfDeposit
NoSecond MessageSent IfCustomerFails
TransactionCompleted IfApproved
(f)

Fig. 4 Comparison of goal models constructed in two methods

scribed above, the difference in the number of goals between the manually constructed and tool-assisted goal models of the ATM system was caused by the goal decomposition granularity. By comparing the two goal models, we found that the difference in the meeting scheduling system stemmed from the same reason.

While the numbers of goals in the tool-assisted goal models are similar between the ATM system and the meeting scheduling system, the total number of questions shows large difference. The difference stems from the structure of the goal models. From Table 2, we can find that the number of the isSubgoal questions is dominant. We can also observe that the number of times of asking the isSubgoal question for the meeting scheduling system is approximately 1.6 times larger than that of the ATM system. Since this question is asked repeatedly until a parent goal is found, the number of times of asking this question depends on the goal model structure, i.e., the structure of the document.

Table 2 also shows that while the times required for constructing the goal model for the ATM system were similar between the manual and tool-assisted construction, the times for the meeting scheduler system were largely different. The difference seems to be caused by the diversity of the system functions, i.e., the diversity of requirements for the system. Since the meeting scheduler has more various functions than the ATM system, the goal model of the meeting scheduler must have various goals to be achieved. In this situation, the examinee needed to carefully decide where to place the goals when manually constructing the goal model. The tool helped the examinee decide the place by asking the relationships between goals.

4.2 Exp 2

Next, we quantitatively compared goal models constructed by six examinees. All examinees were researchers or students in the software engineering field: three of them had experiences of goal modeling (*experienced analysts*), and the others did not have the experiences of goal modeling (*beginning analysts*). All of the examinees constructed goal models, which were for the meeting scheduling system used in Exp 1, using our tool first, then at least one week later, they constructed goal models manually.

Table 3 lists the quantitative results of Exp 2. From the results, even though the numbers of questions were widely different among examinees, the tool-assisted goal modelling was able to make a goal model faster than the manual construction of every examinee. We also found that every set of the goals in the tool-assisted goal model except the one constructed by examinee E was basically the subset of the goals in the manually constructed goal model. Almost all of the additional goals in the manually constructed goal models corresponded to further concretized goals. This phenomenon is caused by the difference in the goal decomposition granularity, described in Sect. 4.1. Moreover, we found that all examinees extracted a similar number of goals in the tool-assisted construction. Examinees E and F extracted relatively small or large number of goals in their manual constructions, respectively. Constructing a goal model based on a certain level of goal granularity is an advantage of using our tool.

Table 2 Number of questions and goals. "MS" stands for the meeting scheduling system.

Question	ATM	MS
askPurpose	1	1
isGoal	31	33
selectType	30	30
isSubgoal	97	156
canAggregateGoals	16	32
askParentGoal	1	0
Total	176	252
Number of goals (manual)	35	35
Number of goals (tool-assisted)	32	31
Elapsed time (manual, minutes)	40	61
Elapsed time (tool-assisted, minutes)	33	17

5. Discussion

The research questions in this evaluation are as follows:

RQ1. Can the proposed process construct correct goal models?

RQ2. To what extent does the proposed process decrease construction cost?

The subsequent subsections answer the research questions in the light of our experiments and discuss the limitations of our process.

5.1 RQ1. Can the Proposed Process Construct Correct Goal Models?

The result obtained from Exp 1 demonstrates that the tool helps to construct a goal model that is similar to a goal model constructed manually. Differences in two goal models, i.e., the manually constructed and tool-assisted goal models, are caused by the following factors: the order of sentences in the given document, and the difference in the goal decomposition granularity. Table 3 shows that the manually constructed goal model contains more goals than the tool-assisted goal model for every examinee except E. The difference in the number of goals comes from the difference in the goal decomposition granularity. This difference stems from the assumption of the tool, i.e., each sentence in the document has at most one goal. An extension of the tool that can remove this assumption will make two goal models

Table 3 Experimental results in Exp 2. Examinees A, B, and C have experiences of goal modeling (experienced analysts). The other examinees D, E, F are engaged in the study of software engineering but do not have the experiences of goal modeling (beginning analysts). The result of the examinee A is the same as the result listed in the column MS in Table 2

Question	A	B	C	D	E	F	Min	Max	Avg.
askPurpose	1	1	1	1	1	1	1	1	1
isGoal	33	33	33	33	33	33	33	33	33
selectType	30	27	28	27	31	27	27	31	28.3
isSubgoal	156	128	163	66	127	72	66	163	118.7
canAggregateGoals	32	20	11	4	30	11	4	32	18
askParentGoal	0	0	0	0	0	0	0	0	0
Total	252	209	236	131	222	144	131	252	199
Number of goals (manual)	35	36	37	36	29	41	29	41	35.7
Number of goals (tool-assisted)	31	28	29	28	32	28	28	32	29.3
Elapsed time (manual, minutes)	61	85	98	68	58	48	48	98	70
Elapsed time (tool-assisted, minutes)	17	26	46	25	31	23	17	46	28

more similar.

The tool prevents analysts from failing to extract goals from a document. The results in Exp 2 show that a beginning analyst examinee E extracted fewer goals in the manual goal model construction than in the tool-assisted goal model construction. This feature helps beginning analysts not miss the goal extraction from the document.

5.2 RQ2. To What Extent Does the Proposed Process Decrease the Construction Cost?

Tables 2 and 3 list quantitative evaluation results. First, comparing the elapsed time between the two construction methods, it can be seen that the proposed process successfully shortened the time required for constructing goal models for both systems. Although the difference is not very significant for the ATM system, the average time of tool-assisted goal model construction for the meeting scheduling system was about 2.5-fold faster than that of the manual construction. If a system has various functions, we expect that the tool helps an analyst construct a goal model more efficiently.

The differences in time between tool-assisted and manual goal modelling in Table 3 show the advantage of our process. Even though further decomposition of goals in the tool-assisted goal model is required to acquire a goal model similar to the manually constructed goal model, it is not a heavy task because the decomposition can be started from specific leaves in the goal model. The tool relieves analysts from the essential burden of deciding the place of various goals, i.e., structuring the goal model.

Despite of the above advantage, we should reduce the number of questions. The results listed in Table 3 show that experienced analysts tend to be asked more questions. The fact that the number of the isSubgoal questions is dominant among the questions explains that experienced analysts carefully choose parent goals of the target goals. In order to further reduce the cost of the goal modeling, we shuold improve the question of finding the parent goal.

5.3 Applicability

The applicability of the proposed process depends on the document given as input data. Therefore, when we deal with a system whose requirements, i.e., features to be implemented, can be explicitly described in the document, we can expect that the process generates a concrete goal model. On the other hand, when we deal with a system such as a cutting-edge system that requires continuous requirements elicitation based on prototyping, it is rather difficult to construct a precise goal model. If a system has various functions, i.e., if an analyst has to deal with various requirements, the proposed process works more efficiently by relieving the analyst from the burden of structuring the goal model.

The structure of the given document also affects the applicability of the proposed process. The current tool works under the assumption that every sentence has at most one goal. Therefore, manual decomposition after the goal model generation is required when sentences in the document have more than one goal. If similar goals are widely distributed in the document, the number of questions will increase.

Even if we use the proposed process, the constructed goal model differs depending on the analyst. When a beginning analyst constructs a goal model using the tool, the analyst can expect that the constructed goal model contains necessary goals. Goal model construction based on a certain level of goal granularity is another advantage of applying our process for the beginning analyst. When an experienced analyst uses the tool, the acquired goal model becomes more similar to the goal model that the analyst would construct manually, by answering more isGoal questions than a beginning analyst. In order to help the experienced analyst more efficiently, further tool/process improvement should be conducted.

6. Threats to Validity

In our experiments, construction of the goal models was performed by six examinees. In general, the quality of goal models depends on the skills of requirements analysts. Therefore, a different result might be obtained if the experiments are performed by different analysts. To mitigate these potential threats to internal validity, we plan to conduct additional experiments where other analysts are involved.

Since the six questions are general and independent from any particular domain, we believe that the proposed method can be applied to other domains different from the domains conducted in the experiments. However, since the numbers of extracted goals in both experiments are not so large, further evaluation concerning to the scalability of the process is desirable.

7. Conclusions

In this paper, we proposed an approach to supporting the construction of goal models. The approach is based on an interactive process using a set of questions. In this process, goals are derived and inserted to the goal model according to the questions and answers. Answers are used to define goals and to find the relationships between goals. We implemented a tool that implements the goal construction process and conducted experiments in two sample systems. The results showed that the constructed goal model is adequately similar to the goal model constructed manually. One of the benefits of using the proposed process is that we can construct a preliminary goal model without comprehending the whole structure of the model.

For future work, we plan to tackle following problems: *goal summarization*, *goal decomposition*, and *implicit goal extraction*. First, the proposed method extracts goals from sentences of a requirements description. The current method requires analysts to summarize the sentences to define goals. Some studies in the natural language processing may help to extract goal descriptions from the sentences. Second, it is sometimes appropriate to decompose these goals into finer goals so that each goal's responsibility can be associated with a single actor. We previously proposed a method of goal decomposition based on extraction rules [5]. We plan to integrate these two methods to tackle the problem of goal decomposition. Third, the current version of the method cannot extract goals that are not explicitly described in the input requirements description. The use of domain knowledge seems to be essential to elicit such goals. Incorporating domain knowledge into our context deserves further studies.

In addition, the questions flow has room for improvement. The current version of the prototype tool asks several questions for each one of the goals. When the number of goals is large, this may result in a burden for the analyst. To mitigate this problem, we plan to extend the proposed method by adding the mechanisms as follows:

- **Text summarization**. Text summarization can be used to replace multiple sentences with a short single sentence, thus, reducing the number of goals extracted.
- **Goal relevance calculation**. This mechanism calculates the mutual relevance between two different goals. The relevance obtained can be used to narrow down the candidates for the parent goal of a new goal.

References

[1] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," Science of Computer Programming, vol.20, no.1-2, pp.3–50, 1993.

[2] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," Proc. 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp.226–235, 1997.

[3] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: a process-oriented approach," IEEE Trans. Softw. Eng., vol.18, no.6, pp.483–497, June 1992.

[4] H. Kaiya, H. Horai, and M. Saeki, "Agora: attributed goal-oriented requirements analysis method," Proc. IEEE Joint International Conference on Requirements Engineering (RE2002), pp.13–22, 2002.

[5] H. Shimada, H. Nakagawa, and T. Tsuchiya, "Constructing a goal model from requirements descriptions based on extraction rules," Proc. Asia Pacific Requirements Engineering Conference (APRES 2017): Requirements Engineering for Internet of Things, vol.809, pp.175–188, 2018.

[6] J. Pimentel, J. Vilela, and J. Castro, "Web tool for goal modelling and statechart derivation," Proc. 23rd IEEE International Requirements Engineering Conference (RE 2015), pp.292–293, Aug. 2015.

[7] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang, "Automated extraction and visualization of quality concerns from requirements specifications," Proc. 22nd IEEE International Requirements Engineering Conference (RE 2014), pp.253–262, Aug. 2014.

[8] T.H. Nguyen, J. Grundy, and M. Almorsy, "Rule-based extraction of goal-use case models from text," Proc. 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015), ESEC/FSE 2015, New York, NY, USA, pp.591–601, ACM, 2015.

[9] X. Franch, G. Grau, E. Mayol, C. Quer, C.P. Ayala, C. Cares, F. Navarrete, M. Haya, and P. Botella, "Systematic construction of i* strategic dependency models for socio-technical systems," International Journal of Software Engineering and Knowledge Engineering (IJSEKE 2007), vol.17, no.1, pp.79–106, 2011.

[10] M. van Zee, F. Bex, and S. Ghanavati, "Rationalization of goal models in GRL using formal argumentation," Proc. 23rd IEEE International Requirements Engineering Conference (RE 2015), pp.220–225, Aug. 2015.

[11] D. Yu, Z. Chen, and Y. Zhang, "From goal models to feature models: A rule-based approach for software product lines," Proc. Asia-Pacific Software Engineering Conference (APSEC 2015), pp.277–284, Dec. 2015.

[12] M. Santos, C. Gralha, M. Goulão, and J. Araújo, "Increasing the semantic transparency of the KAOS goal model concrete syntax," Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, Oct. 22-25, 2018, Proceedings, pp.424–439, 2018.

[13] S. Liaskos, A. Ronse, and M. Zhian, "Assessing the intuitiveness of qualitative contribution relationships in goal models: An exploratory experiment," Proc. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2017), pp.466–471, Nov. 2017.

[14] D. Han, J. Xing, Q. Yang, J. Li, X. Zhang, and Y. Chen, "Integrating goal models and problem frames for requirements analysis of self-adaptive cps," Proc. 41st IEEE Annual Computer Software and Applications Conference (COMPSAC 2017), pp.529–535, July 2017.

[15] L. Piras, E. Paja, P. Giorgini, and J. Mylopoulos, "Goal models for acceptance requirements analysis and gamification design," Conceptual Modeling, Lecture Notes in Computer Science, vol.10650, pp.223–230, Springer International Publishing, Cham, 2017.

[16] C.M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Requirements evolution and evolution requirements with constrained goal models," Proc. International Conference on Conceptual Modeling (ER2016), pp.544–552, 2016.

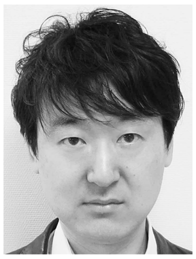
[17] B. Steven, E. Loper, and E. Klein, Natural Language Processing with Python, O'Reilly Media Inc, 2009.

[18] "Graphviz." http://www.graphviz.org/.

[19] A. Cailliau and A. van Lamsweerde, "Integrating exception handling in goal models," Proc. 22nd IEEE International Requirements Engineering Conference (RE 2014), pp.43–52, Aug. 2014.

[20] "Requirements Statement for Example ATM System." http://www.math-cs.gordon.edu/courses/cs211/ATMExample/Requirements.html.

[21] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.



Hiroyuki Nakagawa is currently an Associate Professor at Osaka University. He received his B.S. degree in computer science from Osaka University in 1997, his M.S. degree in computer science from the University of Tokyo in 2007, and his Ph.D. degree in computer science from Waseda University in 2013. He worked in Kajima Corporation from 1997 to 2008. He worked the University of Electro-Communications as an assistant professor from 2008 to 2013. His research interests include models@run.time, self-adaptive systems, requirements engineering, and software evolution.

Hironori Shimada received his M.S. degree in computer science from Osaka University in 2019.

Tatsuhiro Tsuchiya is currently a professor of the Department of Information Systems Engineering at Osaka University. He received the M.E. and Ph.D. degrees from Osaka University in 1995 and 1998, respectively. His research interests are in the areas of model checking, software testing, and distributed fault-tolerant systems.