Elaborating Operational Requirements from Goals

C. Suganya, S. Thirumal

Computer Science & Engineering Department, G.K.M College Of Engineering And Technology.

Assistant Professor, Dept of Computer Science, G.K.M College Of Engineering And Technology.

Abstract— Requirement Engineering comprises of deriving stakeholders goals and their enhancement into operational requirement specification .Insufficiency in the process of any of these task will end up with severe problem in the system development, it becomes expensive to recover. Here will introduce a formal, efficient approach for generating requirements that satisfies the given stakeholder's goals. Goal-based methods have progressively accepted in effect of eliciting, elaborating, analyzing, and specifying software requirements. We use a tool-based framework for combining model checking and inductive learning approach. The model checker applicably validates the goal satisfaction and creates counterexample once incompleteness in the operational requirements are found. The Learner computes the requirements from positive and negative samples. These well-read requirements are reliable with goals. This procedure is done iteratively till no goal destruction is identified.

Keywords- Requirements specifications, elaboration, elicitation methods, operational requirements, model checking, inductive learning.

I. Introduction

Requirement engineering concerned with the elicitation, elaboration, specification, analysis and documentation of goals and requirements to an envisaged system. It infers that organized and repeatable systems should be used to ensure that system requirements are comprehensive and reliable. When the execution of these methods encountered erroneous process, it leads to severe development problem which is hard to repair. This leads to seek automated method for the achievement of goal. This method targets to links the space among high level system goals and low level system requirements.

Goals are the targets, to be attained by the system. The term 'system' here denotes to the software-to-be with the setting. Operational requirements prompt restrictions on the operation to be done by the system. The difficulty in the system development is the expansion of the operational requirements that guarantees the goal. If the manual process is undergone then it will be so problematic and hard to recover. This process lacks automation and generality. Here will present a novel method that uses model checking and inductive learning, is used to detect and correct the incompleteness of a specified set of operational requirements with respect to goals. The particular inductive learning technique we use called Inductive Logic Programming (ILP), Prolog, is given with background knowledge, positive scenarios, negative scenarios are declaratively expressed in temporal logic. The practice of temporal formalism permits computerized investigation and tool enhancement. This makes use of existing knowledge base during inference process. The benefit of this process is inevitably ensures the reliability of well-read requirements with respect to the stakeholders goals.

The projected frame is an iterative procedure involves four segments: In behavior analysis phase, the partial operational specification will be tested against stakeholders goal using Labeled transition system analyzer. If any abnormalities are detected, then desirable and undesirable behaviors will be drawn in the scenario elicitation phase. In the requirement sugges - tion phase, goals, partial operational specification, positive and negative scenarios are used by the prolog system, to compute the operational requirements that guarantees the goal. If the learner creates another set of operational requirements, the manual intervention is required to rate the appropriate requirement that fit for the system function.

A. Problem Statement

The complications in developing operational requirement specification is the expansion of operational requirement that assurances the goals fulfillment. When this process is done manually it leads to time consumption, expensive and error prone. Stakeholders chooses to express their goal through narrative style rather than declaratively in temporal manner, because the scenarios are partial descriptions about specific behavior so it may omit some desired behavior. When the requirements are elicited from scenario-based description is tiresome, and bugs-prone practice, it depends on labor-intensive approach, it can profit from automated approach.

When the process is done manually, they are less accessible to the practitioners. Users become frustrated when the software not meets their needs. Customers who pay for the system may need to pay for the mistakes. when the same system is developed for more than one organization, similar effort have to repeat for every system development. It consumes time and expensive process.

The earlier approaches for requirements elaboration requires more time and cost. Because of fully manual approach, there is likely for the introduction of errors. Lack in the execution of any of the process will leads to development problem, it is more expensive to repair.

B. Objective of The Work

Requirement engineering encompasses various activities from goal elicitation to requirement management. Requirement elicitation is the process of eliciting requirements from the stakeholders, Requirement specification is the process of specifying the requirements to satisfy the stakeholders goal, Requirement analysis is the process of verifying whether the operational requirements satisfies the goal. Requirements are defined at the earlier stage of software development as a de - scription of what should be implemented. The objective is to create an operational specification that is assured to fulfill the goals. The tool-based framework is used to combine model checking and inductive learning approach. The model checking technique is used to check for deviations and inductive learning is used to resolve it. The well-read requirements from the learning system are guaranteed to cover all the desirable behaviors and eliminate negative scenarios.

II. EXISTING SYSTEM

Goal based requirements engineering denotes the usage of goals for requirement elicitation, expansion, organization, description, exploration, negotiation, assignment, documentation, and evolution. One of the major complications in producing a system specification is the elaboration of operational requirements that assurances the fulfillment of the goals. When the process of detecting incompleteness and resolving it, is done manually it leads to erroneous operation. This is principally labor-intensive task and hence is pricey and bugs prone. Very little methodical, laborious support exist, however such methods lacking in required characteristics such as computerization or generalization, making them less accessible to practitioners. This has led investigators to search for rigorous and automatic methods to support the fulfillment of these tasks.

C. Demerits of Existing System

Consumption of more time: If the same system is established for more than one organization, similar efforts have to re - appear for every system development. It leads to more time consumption.

Expensive: When the requirement elaboration process is done manually, it requires more man power, so it is costlier process.

Inaccuracy: There is more human contribution in the manual process, so there is a possibility of likely to have bugs.

III. Proposed System

The projected system will present a innovative approach that uses labeled transition system analyzer as model checker and prolog as inductive learner to identify and resolve incompleteness in the operational requirement specification with re - spect to goal. . The Labeled transition system analyzer legally validates the fulfillment of the goals and generates counterexamples when incompleteness in the operational requirements are found. The prolog system then pick-out operational requirements from the counterexamples and user-provided positive examples. These well-read requirements from the learning system are surefire to entail all the desired behavior and none of the undesired ones. This practice is done iteratively till on no account of goal ruin is discovered.

D. Merits of Proposed System

Less time procedure: The same tool can be recommended for the related system development in various organization.

Limited budget: when tool-based methodology is used there will be less man power , so that price will be reduced.

Reduced error: Because of less labor-intensive, there is a possibility of reduced error.

IV. Proposed System in Detail

E. Analysis Phase

The analysis part, is concerned with inevitably examining whether a given incomplete operational specification entails Fluent Linear Temporal Logic (FLTL) property, may be safety property or progress property.

The Labeled Transition System Analyzer (LTSA) is first used to build Labeled Transition System (LTS) from incomplete operational specification with respect to fluent description. Labeled transition system is verified against goal, its result will be either violation or no violation. The particular event should not have occurred at a particular situation. If any hit from initial state to error state, it destructs the safety property. The trace does not reaches the error state, it satisfies the safety property. Labeled transition system analyzer checks for all terminals in Labeled transition system. If anyone terminals in the event has not look as if, then LTSA is said to disrupt the progress property. The discovery of a destruction trace indicates a absence of requirement for some software-controlled event happening or not happening within the last time unit and hence an incompleteness in the present operational specification.

F. Scenario Elicitation Phase

Identifies the condition on events whose existence or nonexistence in the counterexample indicates to goal destruction. The motive is to yield a result that is applicable to certain problem. The engineer asked to provide positive and negative scenario that illustrates good and bad system behavior. The human intervention is required to detect the event that must not have happened at a certain situation in the trace, that is called negative scenario. The manual-intervention is essential to find the event that should occur at a Specific situation in the trace that is called positive scenario.

As a result the positive and negative scenarios will be elicited.

G. Requirement Suggestion Phase

The incomplete specification, goals, set of fluent definition, elaborated positive and negative scenarios are given as input to the learning system. The result of this level is set of operational requirements, describes an Labeled transition system that admits all the optimistic scenarios and none of the undesirable scenario.

By using Inductive logic programming, we will be provided with the alternative set of requirements.

Prolog System

Machine learning, is the process of constructing a system that can learn from data given by user, which uses logic programming as a uniform representation for examples, background knowledge and hypothesis. Given an encoding of above, Prolog system will develop an theorized logic program which requires all the optimistic examples and none of the undesirable examples. The prolog system will be given with the knowledge bases, then it will compute all well-read operational requirements, that admits all the desirable scenarios and none of the undesirable scenarios.

H. Requirement Selection Phase

Manual Process:

When the learner produces the alternative set of requirements, the labor-intensive is vital to handpick the requirement that fits the system's functionality. The intention behind why only one set is nominated is that, although each set of wellread theories is reliable with the background and elucidates the illustrations, the learning does not assures reliability between the alternate explanations computed in a single iteration, henceforth selecting numerous explanations at once may invalidate truthfulness given in the program. The Labeled transition system model produced from the freshly stretched description does not agree the undesirable situation elaborated in the preceding part and agrees all the optimistic scenarios.

The learning system recognizes common unwanted properties from the negative scenarios.

V. System Architecture

The projected system will present a innovative technique that uses model checker and inductive learner to identify and resolve incompleteness in the operational requirement specification with respect to goal. . The LTSA legally validates the fulfillment of the goals and creates counterexamples when partialness in the operational requirements is detected. The prolog process then generates operational requirements from the counterexamples and human-provided positive examples. These well-read requirements from the learning system are assured to focus all the optimistic scenarios and none of the adverse scenarios. This practice is done iteratively till no goal destruction is detected.

I. Step 1. Analysis Phase

The model checker is first used to build an Labeled Transition System from a incomplete description with respect to explanations . It is then used to validate the LTS in contrast to the goals . The result of the analysis phase is moreover a report that no destruction traces have discovered, in which case the procedure successfully ends, or that a counterexample has detected, in which case it is displayed. The recognition of destruction trace indicates absence of operational requirements.

J. Step 2. Scenario Elicitation Phase

If destruction is detected in the analysis phase, then the engineer needs to elicit positive and negative scenarios that illustrates good and bad behavior of the system. A negative scenario represents an event that must not have happened at certain situation. A positive scenario represents an event that must have happened at specific situation in the trace.

As a result, the desirable and undesirable behavior will be elicited.

K. Step 3. Requirement Suggestion Phase

Prolog technique is process used to derive hypothesized logic program that entails all the positive scenarios and none of the negative scenario.

The knowledge bases, goals, partial operational specification, fluent definitions, and scenarios are translated into a logic program and given to an Prolog system. The result of this level is either true or false , which will be decided based upon the conditions was given by user, each of which permits all the positive scenarios, forbids all the negative ones, and is consistent with the goals and the existing operational specification.

L. Step 4. Requirement Selection Phase

The conditions will be provided with the prolog system, then this will check whether it admits all the desired behavior, or not. If the conditions given by the user agrees with the background knowledge, positive scenarios, goals, then it will provide result as true. Otherwise If the conditions given by the user does not agrees with the background knowledge then it will provide false. From this we have to pick out the missing requirements manually, and make the requirements correct.

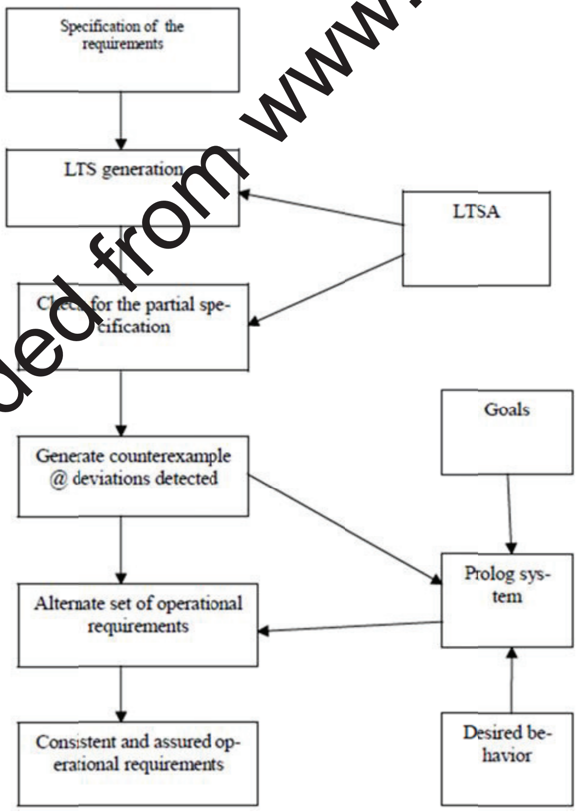
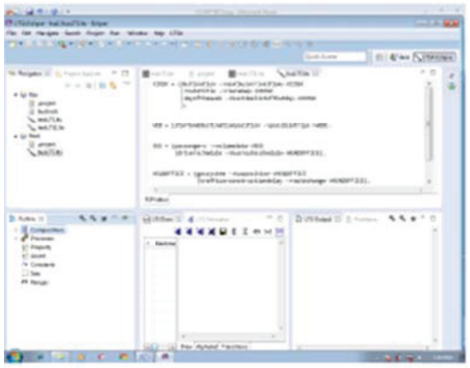

Specification of the requirements
LTS generation
LTSA
Check for the partial specification
Goals
Generate counterexample @ deviations detected
Prolog system
Alternate set of operational requirements
Consistent and assured operational requirements
Desired behavior

Figure 1. System Architecture Diagram

VI. Results



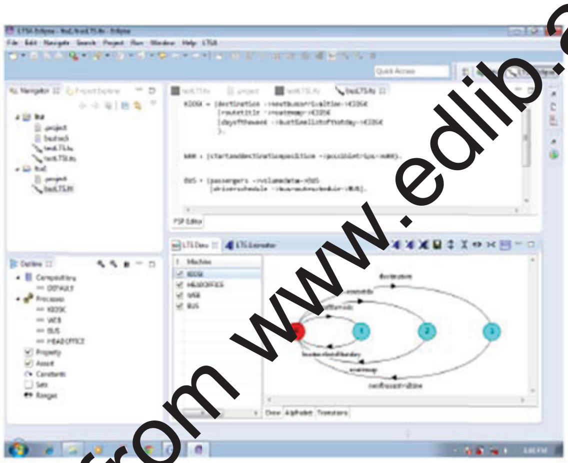
```
KIOSK = (destination ->nextbusarrivaltime->KIOSK
        |routetitle ->routemap->KIOSK
        |daysoftheweek ->bustimelistofthatday->KIOSK
        ).

WEB = (startanddestinationposition ->possibletrips->WEB).

BUS = (passengers ->volumedata->BUS
      |driverschedule ->busroutesschedule->BUS).

HEADOFFICE = [illegible]
        [illegible]
```

Figure 2. Requirement Specification

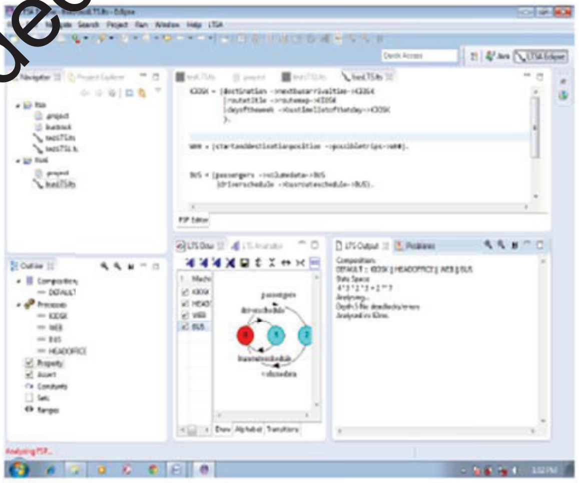


```
KIOSK = (destination ->nextbusarrivaltime->KIOSK
        |routetitle ->routemap->KIOSK
        |daysoftheweek ->bustimelistofthatday->KIOSK
        ).

WEB = (startanddestinationposition ->possibletrips->WEB).

BUS = (passengers ->volumedata->BUS
      |driverschedule ->busroutesschedule->BUS).
```

Figure 3. LTS Generation



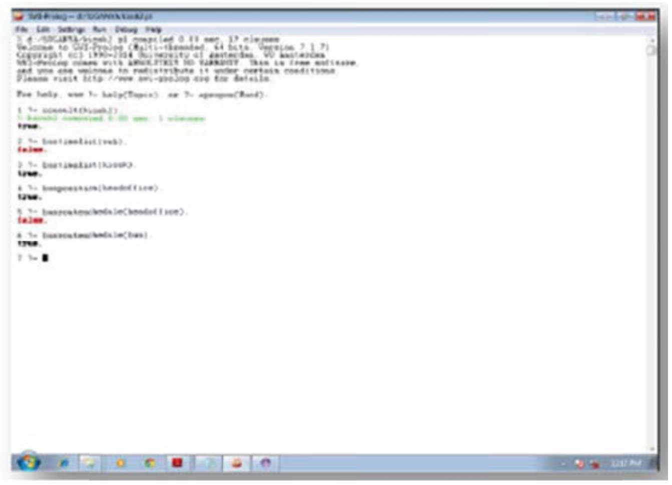
```
KIOSK = (destination ->nextbusarrivaltime->KIOSK
        |routetitle ->routemap->KIOSK
        |daysoftheweek ->bustimelistofthatday->KIOSK
        ).

WEB = (startanddestinationposition ->possibletrips->WEB).

BUS = (passengers ->volumedata->BUS
      |driverschedule ->busroutesschedule->BUS).
```

```
Composition:
DEFAULT = KIOSK || HEADOFFICE || WEB || BUS
State Space:
4*3*2*3 = 2** 7
Analysing...
Depth 5 No deadlocks/errors
Analysed in: 0ms
```

Figure 3. LTS Check



```
[illegible] compiled [illegible] sec, 37 clauses
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.1.7)
Copyright (c) 1990-2014 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- [illegible]
[illegible]
true.

2 ?- [illegible].
false.

3 ?- [illegible].
true.

4 ?- [illegible](headoffice).
true.

5 ?- [illegible](headoffice).
false.

6 ?- [illegible].
true.

7 ?-
```

Figure 4. Requirement Suggestion from Prolog

VII. Conclusion

Requirement elicitation and elaboration is the most important task in requirement engineering. Our focus in this paper is the elaboration of incomplete operational requirement specification by using tool based approach. We have seen that the process of systematically changing requirements specification can be supported by the combination of model checking and Prolog learning are applied respectively. It provides automated, formal support for analyzing the incomplete operational requirements and completing it with respect to goals. The model checking technique is used for detecting the incomplete requirements and Logic programming is used in prolog system for automatically generating the missing requirements that are consistent with the goals. Any requirements are generated automatically guaranteed to satisfy the goal. So this novel method is used to fulfill the operational requirement effectively related with fully labor intensive approach.

VIII. Future Work

For the future, the learning system should be developed with the newer techniques to identify and resolve the incompleteness. This method uses Prolog system to complete the operational requirements, for the future we can use improved and efficient learning technique to automatically generate missing requirements and to improve the completeness in the requirements.

Acknowledgment

My sincere thanks and wishes to acknowledge Dr D. Alrajeh and other contributors for writing many papers regarding requirement elicitation and specification.

References

1. D. Alrajeh, “Requirements Elaboration Using Model Checking and Inductive Learning,” PhD thesis, Imperial College London, 2010.
2. D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel, “Learning Operational Requirements from Goal Models,” Proc. 31st Int’l Conf. Software Eng., pp. 265-275, 2009.
3. D. Alrajeh, O. Ray, A. Russo, and S. Uchitel, “Using Abduction and Induction for Operational Requirements Elaboration,” J. Applied Logic, vol. 7, no. 3, pp. 275-288, 2009.
4. D. Corapi, A. Russo, and E. Lupu, “Inductive Logic Programming as Abductive Search,” Proc. Technical Comm. 26th Int’l Conf. Logic Programming, pp. 54-63, 2010.
5. D. Giannakopoulou and J. Magee, “Fluent Model Checking for Event-Based Systems,” Proc. 11thACM SIGSOFT Symp. Foundations Software Eng., pp. 257-266, 2003.
6. E. Letier and A. van Lamsweerde, “Deriving Operational Software Specifications from System Goals,” Proc. 10th ACM SIGSOFT Symp. Foundations of Software Eng., pp. 119-128, 2002.
7. C. Rolland, G. Grosz, and R. Kla, “Experience with Goal-Scenario Coupling in Requirements Engineering,” Proc. IEEE Int’l Symp. Requirements Eng., pp. 74-81, 1999.
8. AS. D’Avila Garcez, A.Russo, B.Nuseibeh and J. Kramer “Combining Abductive Reasoning and Inductive Learning to Evolve Requirement Specification”.

9. R.A. Kowalski and M. Sergot, "A Logic-Based Calculus of Events" New generation computing, vol. 4, No. 1, pp. 67-95,1986.
10. A. Rifaut, P. Massonet, J. Molderez, C. Ponsard, P. Stadnik, A. Van Lamsweerde, and H. Tran Van, "FAUST: Formal Analysis Using Specification Tools," PROC. 11th IEEE int'l conf. Requirements eng., p. 350,2003.