IRIS-TS: DETECTING INTERACTIONS BETWEEN REQUIREMENTS IN DOORS

Mohamed Shehata[1,3]
Armin Eberlein[2]
Abraham Fapojuwo[1]

[1] Dept. of Electrical & Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada
[2] Dept. of Computer Engineering, American University of Sharjah, PO Box 26666, Sharjah, UAE
[3] Dept. of Information Science, Kuwait University, P.O. Box 5969, Safat, ZIP Code 13060, Kuwait
{Msshehat; Eberlein; Fapojuwo}@ucalgary.ca

Abstract. This paper investigates the problem of requirement interactions which occurs due to negative relationships between requirements when developing software systems. This paper presents IRIS-TS (Requirements Interactions using Semi-formal methods - Tool Support) which identifies and detects requirement interactions using semi-formal methods in any software domain. IRIS-TS is implemented as an independent add-on module that can be added to DOORS (which is one of the most famous and commonly used requirements management tools). This paper presents also a case study in which the proposed IRIS-TS approach was successfully used as an add-on module in DOORS to detect interactions between smart homes requirements which represent a new application domain for interaction detection. The presented case study is the first comprehensive effort to fully detect interactions in the smart homes domain.

Keywords: Requirement Interactions, Requirements Managment, DOORS

(Received January 27, 2006 / Accepted July 06, 2006)

1. Introduction

Studies have claimed that in order to succeed in developing high-quality software systems, it is necessary to have correct and unambiguous requirements [1]. This makes requirements engineering (RE) a vital part of software development [2-5] and critical to the success of the entire project. A key issue in obtaining a set of clear requirements is how to manage negative relationships between requirements [6] [7]. Robinson *et al.* in [8] defines requirements interactions management as "the set of activities directed towards the discovery, management, and disposition of critical relationships among a set of requirements". Requirements often interact when developing new systems because of the heterogeneity and diversity of stakeholders [8] or because of reusing already existing requirements from previous similar projects where people make the assumption that the reused requirements will increase safety because they have been exercised extensively [9]. In either case, developing a software project should be done with an ongoing effort to discover and resolve interactions that arise between requirements. A review of the current practice of interaction detection showed that there are two extremes: one extreme uses informal detection approaches using domain experts who rely on their experience with no systematic approach to follow. The other extreme uses formal approaches, such as the Specification and Description Language SDL [10]. However, domain experts are expensive, hard to find and prone to errors [11]. Many formal interaction detection approaches, which can be found in [12-18], provide fairly accurate detection of interactions. But not every company has the time and resources necessary to carry out a formal verification of their systems under development. For example, many domains like commercial PC software do not use formal models to validate their software but rather have an expert who identifies critical interactions. This is because the consequences of interactions are relatively minor.

This paper proposes the use of semi-formal techniques for detecting requirements interactions by using IRIS-TS (Identifying Requirements Interactions using Semi-formal methods - Tool Support). The semi-formality of IRIS-TS is achieved through the use of tables, graphs, interaction detection scenarios, and subjective detection to detect interactions. This requires visual system representation and does not require extensive mathematical modeling of the system under investigation as opposed to formal methods and at the same time IRIS-TS would provide a well structured and systematic approach for interaction detection as opposed to human experts who have no systematic way for detecting interactions and relay only on their expertise.

To enrich IRIS-TS powerfulness, IRIS-TS is implemented as an independent add-on module that can be added to DOORS [19], which is a very commonly used requirements management tool, to facilitate requirements interaction detection in real life projects.

As a proof of theory, the proposed IRIS-TS was successfully used as an add-on module to DOORS and was applied to detect interactions between smart homes requirements which is a new application domain for interaction detection. The obtained results show that IRIS-TS was able to detect 83 interactions among 35 requirements using 525 pair-wise comparisons as opposed to 630 a human expert would have to do using only experience.

The remainder of this paper is structured as follows: Section 2 provides details on the theory used in IRIS-TS for detecting interactions. Section 3 provides the implementation of IRIS-TS and how a prototype of IRIS-TS was developed in DOORS using the DXL programming language. Section 4 details a case study that was conducted using IRIS-TS for detecting interactions between smart homes requirements and provides a summary of the obtained results. Finally Section 5 draws the paper conclusions.

2. Theory Behind IRIS-TS

2.1 Internal Structure of IRIS-TS

IRIS-TS is the outcome of research into the application of light-weight semi-formal approaches to detect interactions in any domain in a cost effective manner.

IRIS-TS is a systematic six step procedure that produces tables and graphs in the first five steps and then detects interactions in the last step using the tables and graphs created in the previous steps. The six steps of IRIS-TS

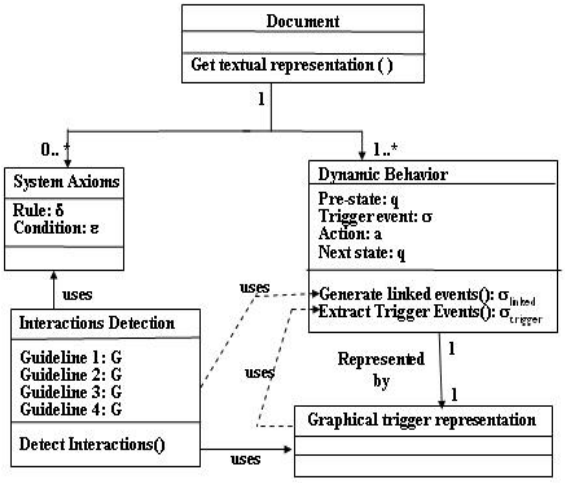

Figure 1: Class Model of IRIS-TS

are ordered in such a manner that this translation of requirements into graphical and tabular representations is gradually achieved. The objective of these representations is to facilitate the application of the interaction detection guidelines. The six step procedure of IRIS-TS can be represented using the model shown in Figure 1. The following describes IRIS-TS six steps:

- Step 1: Requirements classification: The Requirements are classified into one of the following two categories: *System Axiom Requirements* and *Dynamic Behavior Requirements*. A requirement is considered to be a system axiom requirement if it describes a property that has to be preserved at all times. For example a system axiom might state "The system shall maintain the temperature of the water from the hot water tap at 45 ^{0}C". On the other hand, a requirement is considered as a dynamic behavior requirement if it specifies the reaction of the system when a certain event occurs. For example, a dynamic behavior requirement states "The system shall open the bedroom curtains at 7:00 am every morning".
- Step 2: Requirements attributes identification: This step is aimed at identifying different attributes within the system requirements. To represent these attributes, two tables are generated:
  - The first table is used to represent system axiom requirements using attributes *Rules, Conditions*
  - The second table is used to represent system dynamic behavior requirements using the four attributes *Pre-state*, *Trigger event*, *Action*, and *Next state*

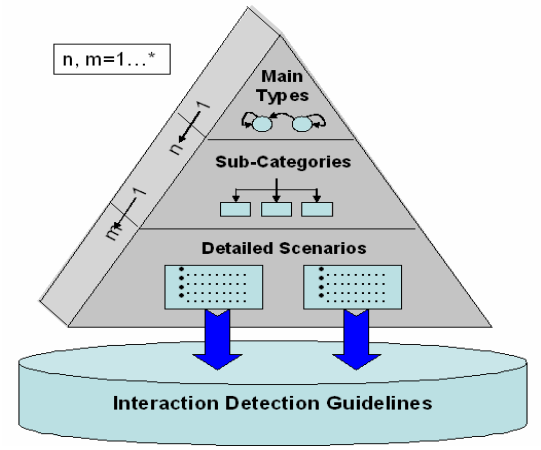

Figure 2: A general interaction detection taxonomy

This step is done by identifying the different attributes of the requirements from the textual description of the requirements and recording them in the system axiom and dynamic behavior tables.

- Step 3: Trigger events extraction: This step identifies and extracts all the different trigger events from the dynamic system behavior requirements. An event is called a trigger event of a requirement, if when it occurs will trigger the requirement to execute the action specified in its textual description. This step is applied only to dynamic behavior requirements.
- Step 4: Linked events identification: During this step, a table is developed that contains all linked trigger events. A trigger event is called a linked trigger event if it can lead to the occurrence of other trigger events. For example, the event E1 "Windows opened" is linked to the event E2 "Temperature changed".
- Step 5: Graphical trigger representation: In this step an event-based graphical representation is used to represent each event with requirements it triggers. This graphical representation will later facilitate interaction detection between requirements.
- Step 6: Interactions detection: During this step, the developer detects interactions between requirements using the IRIS-TS interaction detection guidelines (see section 2.2). This step applies to all requirements.

2.2 General Interaction Taxonomy

The actual detection of interaction takes place during the last step of IRIS-TS. A human developer uses the tables and graphs generated during the earlier steps to detect interactions using specific detection guidelines. These guidelines are designed for use by non-experts thus reducing the interaction detection cost. These guidelines are domain independent, which means that they can be applied in any domain including the smart homes domain selected as case study in this paper.

In order to develop interaction detection guidelines, a general interaction taxonomy had first to be established. Therefore, an extensive review of currently existing definitions of interactions was carried out. Based on this review, a three layer taxonomy was developed as shown in Figure 2. The first layer of the taxonomy contains the main types of interactions. In the second layer of the taxonomy these main types are decomposed into subcategories. The third layer of the taxonomy associates each subcategory of layer 2 with one or more scenarios that describe an interaction situation. Each of these scenarios generated a guideline on when two requirements interact.

Four guidelines from the taxonomy were used in the smart home case study. The following contains a description of these guidelines:

1. Interaction between two system axioms requirements: Guideline 1 states that "*Two system axiom requirements interact when the rule attribute of one system axiom requirement contradicts the rule attribute of a second system axiom requirement*". For example, assume the following scenario: A system axiom requirement states "Web pages that are used to submit confidential information will always require secure logon and secure communication with the web server" Another system axiom requirement states "The transition time from any page to the transactions page should be minimal and should at all times be less than 10 seconds". If the security of the logon to the transaction page is done using a username/ password, then there is an interaction.
2. Interaction between a system axiom requirement and a dynamic behavior requirement: Guideline 2 states that "*a system axiom requirement interacts with a dynamic behavior requirement when the action attribute of the dynamic behavior requirement contradict the rule attribute of the system axiom requirement*". For example: A dynamic behavior requirement states "When the lift is overloaded, doors will not close". Also consider the following system axiom requirement "The lift shall always serve unserved calls".
3. Interaction between two dynamic requirements

   3.1. Guideline 3 states that "*two dynamic behavior requirements interact if:*

1. *The Previous State attributes of both requirements are the same, AND*
2. *The Trigger Event attributes of both requirements are the same, AND*
3. *The Action attributes of both requirements contradict each other"*

To illustrate this interaction assume the following scenario: one dynamic requirement activates a remote access module if there is an incoming call with no answer for 6 rings while the second dynamic requirement activates an answer machine if there is an incoming call with no answer for 6 rings. Obviously the system will not be able to do both actions simultaneously leading to interaction.

3.2. Guideline 4 states that "*two dynamic requirements will interact when they are triggered by linked events and the action attribute of one requirement will cancel or contradict the action attribute of the other requirement*". For example, assume the following scenario: one requirement lets a CD player play music for three hours starting 7:00 pm and the other requirement shuts down all audio/video devices after 9:00 pm. Obviously the second requirement will cancel the action of the first requirement before its completion.

3. Implementation of IRIS-TS

IRIS-TS is implemented as independent code files that can be inserted as an add-on to DOORS [19] to facilitate the detection of requirements interactions. DOORS is one of the most commonly used requirements management tools for documenting and managing requirements for software systems.

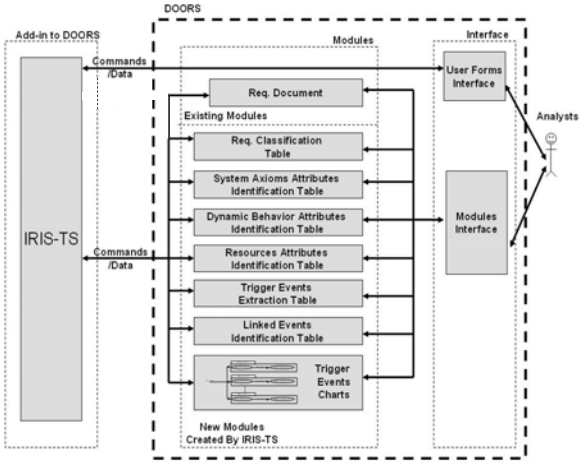

Figure 3: IRIS-TS as an add-on module to DOORS

However, DOORS does not have any sort of interaction detection support built in it. IRIS-TS is implemented to be installed as an add-on to extend DOORS to support requirements interaction detection.

DOORS consists of modules that contain data and interfaces. A module is the way that DOORS uses to store data. A module is like a sheet on which data is written and stored. For example, in a specific software system that uses DOORS, there will be a module that contains all the system requirements and a module that contains all the tests for validating the final product. Each module consists of Objects and Attributes which corresponds to rows and columns, respectively. Objects and attributes are used to represent the information stored within a module. For example, the requirements module will contain an object (row) O1 that represents a requirement R1. The object O1 has attributes that describe requirement R1 such as ID, Object text, and Created by. These attributes are part of the default set of attributes that comes with DOORS.

The concept of attributes used in IRIS-TS is achieved using the DOORS concept of attributes, only this time new customized attributes are created in DOORS through the IRIS-TS code. For example, IRIS-TS when executed will create new attributes that are applicable for all modules of DOORS such as "PreState", "Action", and "NextState". The concept of modules has been used by IRIS-TS to represent the tables and graphs that are generated through the different steps of IRIS-TS. For example, IRIS-TS will create a module called "Trigger Events Extraction Table" to correspond to the table created in the trigger events extraction step. The diagram in Figure 3 shows how IRIS-TS is implemented as an add-on module to be added to DOORS.

The IRIS-TS tool was programmed using DOORS programming language DXL (DOORS eXtension Language). The programming language DXL is a scripting language specially developed for DOORS. DXL can be used provide many features, such as file format importers and exporters, impact and traceability analysis and inter-module linking tools. DXL can also be used to develop larger add-on packages such as IRIS-TS. The DXL language is based on an underlying programming language whose fundamental data types, functions and syntax are largely based on C and C++.

The way DXL is used is to either enter individual scripts in a specific window in DOORS and run these scripts to see how they work, or the other alternative would be to

develop an add-on package that can be added to DOORS and with some specific scripts the DXL script can appear as a menu on the top bar of DOORS. The latter way was used in implementing IRIS-TS to be as a menu option in DOORS as shown in Figure 4.

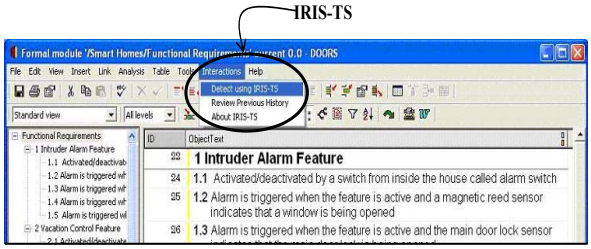

Figure 4: IRIS-TS as drop-down menu in DOORS

4. Detecting Smart Home Interaction using IRIS-TS

This section presents the application of IRIS-TS to detect interactions between smart homes requirements. For each step of IRIS-TS, two screen shots are presented. The first screenshot shows how IRIS-TS performs the step being executed. The second screenshot shows the result that IRIS-TS has generated. The requirements of the smart homes are stored as objects. Each requirement will have an ID attribute to identify it and an object text attribute that contains the textual description of the requirement.

4.1 Smart Homes Requirements

IRIS-TS was applied to detect interactions between 35 requirements that are described as follows:

Intruder Alarm Requirements

P1.1: Activated/deactivated by a switch from inside the house called alarm switch.

P1.2: Alarm is triggered when the feature is active and a magnetic reed sensor indicates that a window is being opened

P1.3 Alarm is triggered when the feature is active and the main door lock sensor indicates that the main door lock is being opened

P1.4 Alarm is triggered when feature is active and a PIR sensor indicates movement in X1 (where X1 is a location)

P1.5 Alarm is triggered when the feature is active and pressure pads indicate the presence of a person in X2 (X2 is a location)

Vacation Control Requirements

P2.1 Activated/deactivated by a switch from inside the house called vacation switch.

P2.2 Turns on TV for 60 min. at X3 (where X3 is time)

P2.3 Turns on lights for 60 minutes at X4 in X5, (where X4 is a time and X5 is a location)

Main Door Control Requirements

P3.1 Locks the main door lock of the house when the main door is shut.

P3.2 Occupants can unlock and open the main door from inside by interior switch

P3.3 Unlocks and opens the main door when the Gas/Heat/Smoke sensor is triggered.

Audio/Visual Control Requirements

P4.1 Occupants can control all A/V devices through remote controls

P4.2 Turns on/off X6 A/V device at X7, (where X6 is an A/V device and X7 is a time)

In home Audio Levels Control Requirements

P5.1 Presets the audio level of audio device X8 to X9 when turned on, (where X8 is an A/V device and X9 is an audio level)

P5.2 Occupants can set X10 as a maximum audio level throughout the house, (where X10 is an audio level)

Heating, Ventilation and Air Conditioning Control Requirements

P6.1 Increases/decreases the ambient temperature inside the house to X11 when the readings from the thermostats are different from this preset temperature, (where X11 is a temperature)

P6.2 Increases/decreases the temperature of the house to X12 at X13, (where X12 is a temp. and X13 is a time)

Water Temperature Control Requirements

P7.1 Maintains the temperature of the hot water from the hot water tap in the kitchen at 45 degree centigrade.

P7.2 Maintains the temperature of the hot water from the hot water tap of the bathroom at 40 degrees.

Lights Control Requirements

P8.1 Increases/decreases the light intensity to correspond to the increase/decrease of a light dimmer.

P8.2 Increases the light intensity during night in X14 to the maximum within 2 minutes when a positive PIR signal is received from X14, (where X14 is a location)

P8.3 Automatically shuts down the lights during night in X15 when a PIR signal is negative for 15 minutes from X15, (where X15 is a location)

P8.4 Automatically turns on the lights according to a daylight sensor when the night begins.

Curtains and Blinds Control Requirements
P9.1 Automatically opens/closes the curtains and blinds in X16 at X17, (where X16 is location and X17 is time)
P9.2 Automatically opens/closes the curtains/blinds in X18 according to daylight sensor, (X18 is a location)

Windows Control Requirements
P10.1 Opens/closes the windows in X19 at X20, (where X19 is a location and X20 is a time)

Water Overflow Control Requirements
P11.1 Closes the water tap when the water reaches or exceeds 75% of the total volume of the sink or the tub either in the kitchen or in the bathroom

Remote Access Requirements
P12.1 Activates a remote access module when an incoming telephone call has not been answered within X21 rings, (where X21 is number of phone rings)

Telephone communication Requirements
P13.1 Enforces the presence of a telephone line with either standard POTS or VOIP
P13.2 Activates an answer machine to record messages when receiving a call with no answer for X22 rings, (where X22 is number of phone rings)

Stove Control Requirements
P14.1 Shut down and prevent any activation of the stove during X23 and X24, (where X22 and X23 are time)

Fan Control Requirements
P15.1 Automatically turns on the kitchen fan when the humidity sensor is triggered
P15.2 Automatically switches off the kitchen fan when humidity signal is lost for 20 min. while the fan is on.

Control of Various Appliances Requirements
P16.1 Occupants can control various appliances like the food processor, water boiler…etc. using remote controls

4.2 Results applying IRIS-TS to Smart Homes Requirements

4.2.1 Results of Step 1: Requirements Classification

The first step of IRIS-TS is requirements classification into system axioms or dynamic behaviour requirements. This step is carried out as shown in Figure 5. IRIS-TS will display a message for each requirement and ask the analyst to classify it as a system axiom or a dynamic behaviour requirement.

Once finished displaying all requirements to be classified to the analyst, IRIS-TS will create a new module to correspond to the requirements classification table created in IRIS-TS step 1. The created module contains all the requirements along with their classification stored in an attribute called classification. The result of this table is shown in Figure 6.

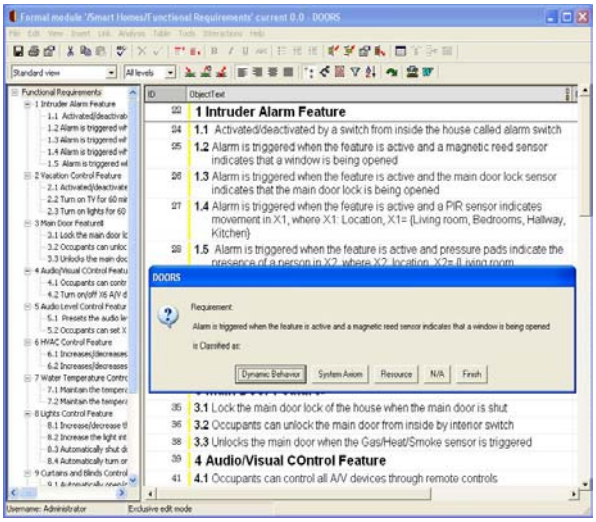

Figure 5: Performing Requirements classification (Step 1)

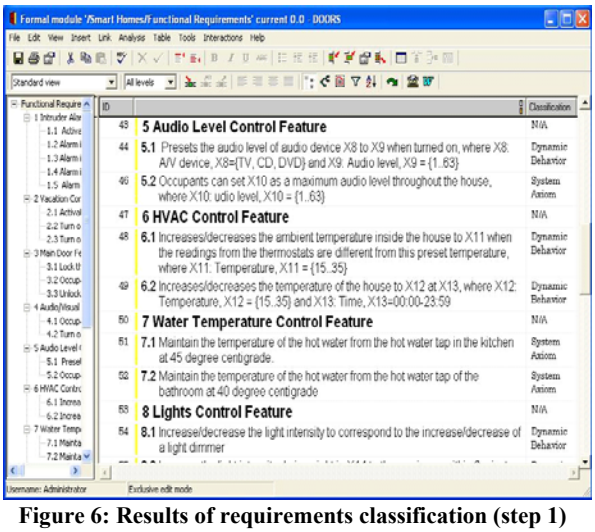

Figure 6: Results of requirements classification (step 1)

4.2.2 Results of Step 2: Requirements Attributes Identification

The second step of IRIS-TS is attributes identification for all system requirements (both axioms and dynamic). In the smart homes case study, IRIS-TS will start displaying messages to the analyst asking to identify the

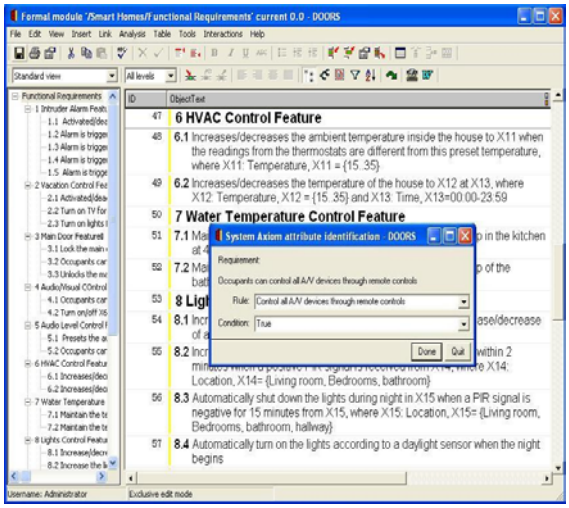

Figure 7: Performing system axioms attributes identification (step 2)

attributes of the system axioms first (Rule and Condition) and then to identify the attributes of the dynamic behaviour requirements (Prestate, Trigger Event, Action, and Next State). Note that the attributes ID and Description are obtained automatically from the original requirement module.

For the sake of aiding the analyst to be consistent in using the same terminologies for defining the attributes, a drop-down buffer is available with each attribute to list all previously entered attributes.

Once IRIS-TS finishes all requirements attributes identification, it will create two new modules with attributes corresponding the requirements attributes to save all the data collected from the analyst.

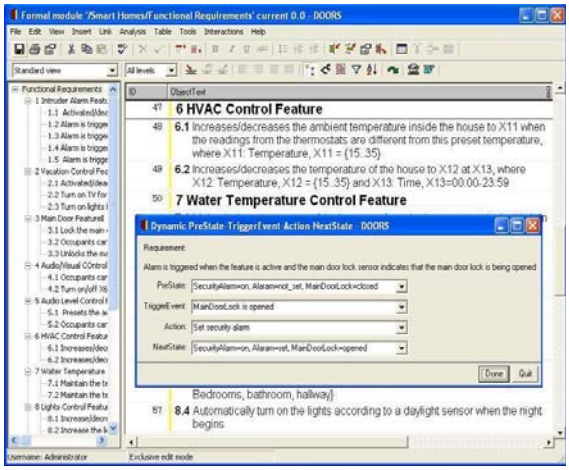

Figure 8: Performing dynamic behaviour attributes identification (step 2)

Figures 7 and 8 shows screenshots for messages to the analyst to enter values for the system axiom and dynamic behaviour requirements attributes respectively. Figures 9 and 10 shows screenshots for the created modules for the system axiom attributes identification and dynamic behaviour attributes identification.

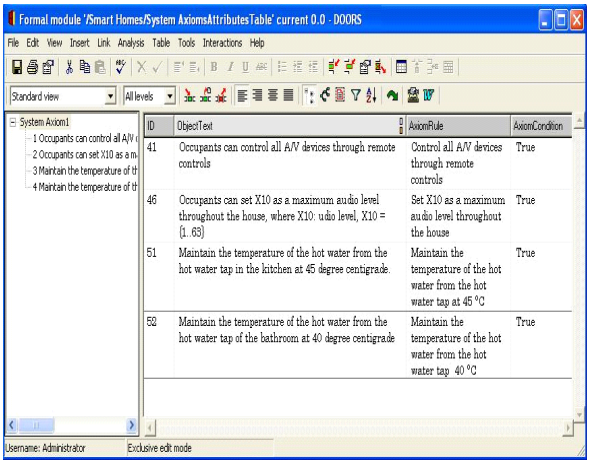

Figure 9: Results of system axioms attributes identification (step 2 in IRIS-TS)

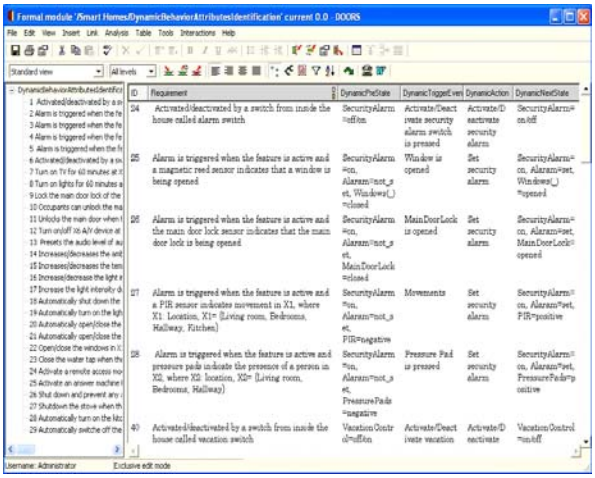

Figure 10: Results of dynamic behaviour attributes identification (step 2)

4.2.3 Results of Step 3: Trigger Events Extraction

The Trigger events extraction step is automatically done with no input from the analyst. IRIS-TS will examine the dynamic behaviour attributes identification (Figure 10) and automatically extracts all unique trigger events and links them to the requirements they trigger. After that, IRIS-TS will create a new module to correspond for the trigger events extraction table. The created module will contain all unique trigger events and the requirements that each trigger event triggers (Figure11).

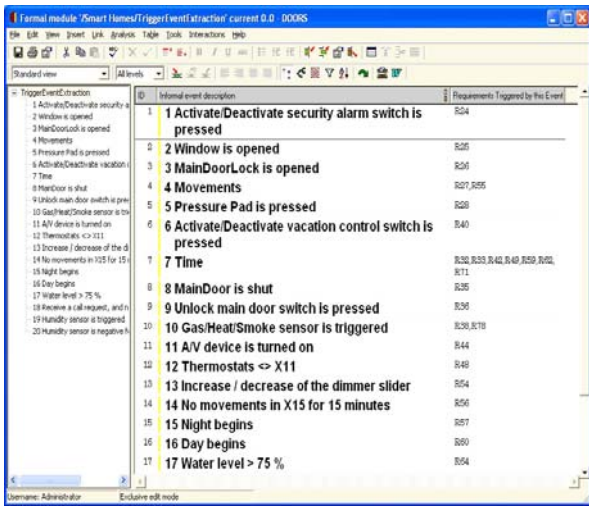

Figure 11: Results of trigger events extraction (step 3 of IRIS-TS)

4.2.4 Results of Step 4: Linked Events Identification

The linked events extraction step is performed by having IRIS-TS examining the trigger events module that was created in step 3 (Figure 11). Then, IRIS-TS displays messages to the analyst asking him to determine if the event under investigation is linked to other events. The analyst can choose from a drop down menu that contains all other available events and the analyst can choose as many as s/he wants. Once IRIS-TS finishes receiving input from the analyst, it will create a module that corresponds to the linked events table. Figure 12 shows the execution of the linked events identification while Figure 13 shows the created linked events module by IRIS-TS.

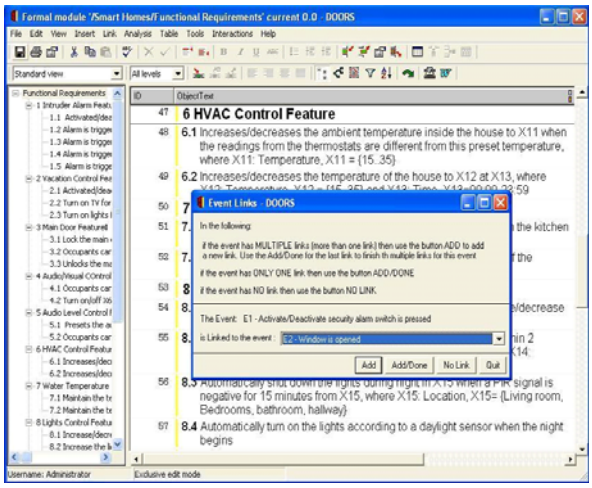

Figure 12: Performing linked events identification

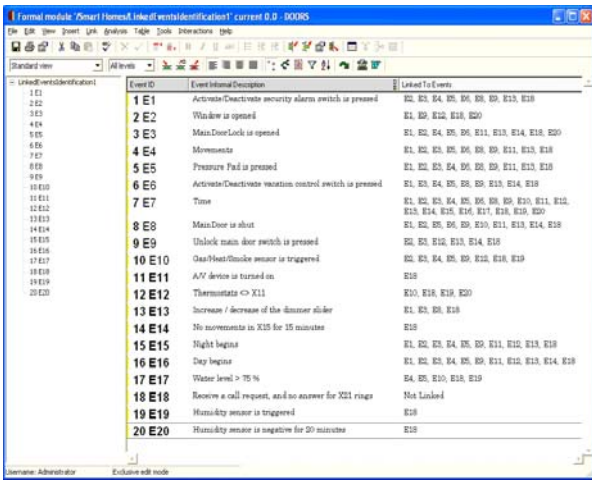

Figure 13: Results of linked events identification (Step 4)

4.2.5 Results of Step 5: Graphical Trigger Representation

The 5th step in IRIS-TS is done automatically without any input from the developer. This step generates trigger events charts and saves them in a module called trigger events charts module. Figure 14 shows a sample of the generated trigger event charts for event E4. It is worth saying that in Figure 14 the button Toggle Length will display the complete text in the diagram or, when repressed, will display a clipped portion of the text to provide uniform non-overlapping display.

Figure 14: Graphical Trigger Representation

4.2.6 Results of Step 6: Interaction Detection

In this step, the developer detects interactions between requirements using the developed tables and graphs generated in the previous 5 steps of IRIS-TS with the help of the guidelines introduced in section 2.2. The detection done here has a subjective part which is the final judgment from the analyst on two requirements if they interact or not. A summary of the obtained results is presented in Table 1. The requirement column contains the requirement under investigation while the

interacting requirements column lists the requirements that interact with the requirement under investigation. Note that when an interaction between two requirements is detected (e.g. P1.1 and P1.2), then this interaction is listed in the row of the first requirement (P1.1) only and will not be repeated as part of the interactions of the second requirement (P1.2).

Table 1: Summary of interaction detection results

Req.	Interacting Requirements
P1.1	P1.2, P1.3, P1.4, P1.5, P3.2, P8.2, P10.1, P12.1
P1.2	P3.2, P10.1, P12.1
P1.3	P3.2, P3.3, P12.1
P1.4	P3.2, P8.2, P12.1
P1.5	P3.2, P8.2, P12.1
P2.1	P2.2, P2.3, P10.1, P12.1
P2.2	P4.1, P4.2, P5.2, P9.1, P9.2, P10.1, P12.1
P2.3	P6.1, P6.2, P8.1, P8.2, P8.3, P8.4, P9.1, P9.2, P10.1, P12.1
P3.1	P3.3, P12.1
P3.2	P6.1, P6.2, P12.1
P3.3	P6.1, P6.2, P12.1
P4.1	P4.2, P5.1, P5.2, P12.1
P4.2	P5.2, P12.1
P5.1	P5.2, P12.1
P5.2	P12.1, P16.1
P6.1	P6.2, P10.1, P12.1
P6.2	P10.1, P12.1
P7.1	Nothing
P7.2	Nothing
P8.1	P8.2, P8.4, P12.1
P8.2	P12.1
P8.3	P12.1
P8.4	P9.1, P9.2, P12.1
P9.1	P9.2, P12.1
P9.2	P12.1
P10.1	P12.1
P11.1	P12.1
P12.1	P13.1, P13.2, P14.1, P14.2, P15.1, P15.2, P16.1
P13.1	Nothing
P13.2	Nothing
P14.1	P16.1
P14.2	P16.1
P15.1	P16.1
P15.2	P16.1

4.2.7 Discussion of the Obtained Results

This section evaluates and discusses the obtained results using the three measures of *Suitability, Accuracy,* and *Reduction.* Suitability measures how suitable the approach was for detecting interactions between the 35 requirements. This procedure is most beneficial for systems that have many requirements describing the dynamic behavior versus a few requirements describing system axioms as in such a case there is a significant reduction of comparisons (see below). Since the smart home case study had only 6 system axioms versus 29 dynamic behavior requirements, the procedure was very suitable for this particular case study.

Accuracy shows how precise the process of detecting interactions was and if any feature interactions were missed. Unfortunately, there are no fully documented results in the literature with which we could have compared our results. However, Kolberg *et al.* in [20] lists an overview of the interactions that arise between services supporting networked appliances in a smart home environment. This overview included some interaction examples. All interaction examples mentioned in [20] were detected using IRIS-TS.

Reduction is a measure that indicates how much the necessary number of pair-wise comparisons could be reduced due to the application of IRIS-TS. The number of comparisons that were done using IRIS-TS compared to the number of comparisons that an expert would have to do was 525 comparisons as opposed to 630. Thus we have achieved a 16.7% reduction in the number of pair-wise comparisons. Although this reduction cannot be directly translated into the same percentage reduction in time and effort due to the fact that the application of IRIS-TS will cause an overhead in time and effort, but we claim that IRIS-TS, as a structured approach, is likely to increase the number of detected interactions. Hopefully, the increase in the number of detected interactions will compensate for the additional time and effort overhead of applying IRIS-TS.

5. Conclusions

This paper proposed the use of IRIS-TS which is a semi-formal approach as a cost-effective way for detecting requirement interactions in any domain. IRIS-TS is implemented as an independent add-on module that can be added to DOORS (which is one of the most famous and commonly used requirements management tools) to facilitate requirements interaction detection in real life projects. A case study was carried out for detecting interactions among requirements for a smart home and was presented in this paper. This case study showed the effectiveness of using the proposed semi-formal IRIS-TS approach for detecting interactions among requirements. IRIS-TS achieved a fairly accurate detection of interactions with a reduction of 16.7% in time and effort.

References

[1] J. O. Palmer and N. A. Fields, "An Integrated Environment for Requirements Engineering," *IEEE Software*, vol. 9, pp. 80-85, 1992.

[2] I. Bray, *An introduction to requirements engineering*. Harlow: Addison-Wesley, 2002.

[3] J. A. Goguen and M. Jirotka, *Requirements engineering : social and technical issues*. London: Academic Press, 1994.

[4] E. Hull, K. Jackson, and J. Dick, *Requirements engineering*. London: Springer, 2002.

[5] I. Sommerville and P. Sawyer, *Requirements engineering : a good practice guide*. Chichester, Eng. ; New York: Wiley, 1999.

[6] L. Jiang, A. Eberlein, and B. H. Far, "A Methodology for RE Process Development," presented at 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS), Czech Republic, 24-27 May 2004.

[7] M. Shehata, A. Eberlein, and J. Hoover, "Requirements Reuse and Feature Interaction Management," presented at 15th International Conference on Software & Systems Engineering and their Applications (ICSSEA'02), Paris, France, December 3-5, 2002.

[8] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Computing Surveys (CSUR)*, vol. 35, pp. 132-190, June 2003.

[9] N. G. Leveson, *Safeware, System Safety, and Computers*: Addison-Wesley Pub. Co. Inc., 1995.

[10] J. Ellsberger, A. Sarma, and D. Hogrefe, *SDL : formal object-oriented language for communicating systems*, 2. ed. London: Prentice Hall, 1997.

[11] B. Boehm and H. In, "Identifying quality-requirement conflicts," presented at Proceedings of the Second International Conference on Requirements Engineering, Los Alamitos, CA, USA, 1996.

[12] N. Griffeth and Y.-J. Lin, *Feature Interactions in Telecommunications Systems*. St. Petersburg, Florida, USA: IOS Press Inc., 1992.

[13] L. G. Bouma, H. Velthuijsen, and IEEE Communications Society, *Feature interactions in telecommunications systems*. Amsterdam: IOS Press, 1994.

[14] K. E. Cheng and T. Ohta, *Feature Interactions in Telecomunications III*. Kyoto, Japan: IOS Press Inc., 1995.

[15] K. Kimbler and L. G. Bouma, *Feature Interactions in Telecommunications and Software Systems V*. Lund, Sweden: IOS Press, 1998.

[16] M. Calder and E. Magill, *Feature Interactions in Telecommunications and Software Systems VI*. Glasgow, Scotland: IOS Press Inc., 2000.

[17] D. Amyot, *Feature Interactions in Telecommunications and Software Systems VII*. Ottawa, Canada: IOS Press Inc, 2003.

[18] P. Dini, R. Boutaba, and L. Logrippo, *Feature Interactions in Telecommunications Networks*. Montreal, Canada: IOS Press Inc,, 1997.

[19] "Telelogic DOORS", http://www.telelogic.com/

[20] M. Kolberg, E. H. Magill, and M. Wilson, "Compatibility Issues between Services Supporting Networked Appliances," *IEEE Communications Magazine*, vol. 41, pp. 136 - 147, 2003.