2020

A multi-layered desires based framework to detect evolving non-functional requirements of users

Peng Sun
Iowa State University

A multi-layered desires based framework

to detect evolving non-functional requirements of users

by

Peng Sun

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Carl K. Chang, Major Professor
Hridesh Rajan
Simanta Mitra
Jin Tian
Pavan Aduri

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2021

DEDICATION

I dedicate this work and give special thanks to my academic advisor Carl Chang for helping me develop my research skill and smooth out many impediments to this study.

I would like to dedicate this thesis to my family without whose support I would not have been able to complete this work.

I also dedicate this thesis to my many friends who have supported me throughout the entire doctoral program, especially Jingwei Yang for offering much help on my study and completing of the thesis, and Yunfei Feng for so many happy moments spent together.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to my major professor Carl K. Chang, and my committee members, Drs. Rajan, Mitra, Aduri, and Tian, for their instruction and inspiration on my research.

I would additionally like to thank my friends, department faculty and staff for making me enjoyed an amazing time in Iowa State University. Also, I appreciate the help from those students who were willing to participate in my research studies.

Many thanks to Laurel Ming and Tamara Schmeisser for editing and proofreading this thesis and for being good friends.

ABSTRACT

Non-Functional Requirements (NFRs) have a great impact on all the downstream activities in a software's entire life-cycle. It is important to capture newly emerged NFRs for system evolution. NFRs are treated as quality goals in Goal Oriented Requirements Engineering. Under the concept of the goal, the interrelations among users, system agents, and scenarios are complex, so the quality factors related to users' concerns cannot be easily studied. In this study we distill human factor from the concept of goal and deliberately represent NFRs through contribution relations among human desires. By doing so, NFRs can be better exploited from humans' perspective. We noticed that recent work in the area of requirements engineering shows that through the combinational use of goal inference, user behavioral and system contextual data functional requirement can be elicited in the form of task-level alternative features. Our basic assumption is that there is a chance to extract new NFRs from users' mental states, specifically their desires, because the concepts of goal and desire are closely connected. A statistical model under the Situ framework is proposed to infer human desires of multiple levels of abstraction from contextual data. In our multi-layered desire inference method, we consider inference results and try to make sense of results with different levels of inference confidence. We ran a case study to show how to elicit users' new NFRs in three cases from the contributing relations among desires in different abstraction levels. To bridge desires and goals in requirements engineering and apply the elicited NFRs on the practical system evolution, we extended the i* framework to build the dependency relations between desire contributing relations and system tasks. Several implications of this work are also discussed.

Keywords – Conditional Random Fields, Desires Inference, Nonfunctional Requirement, Software Engineering

CHAPTER 1. INTRODUCTION

"Nonfunctional requirements are properties the functionality must have." [51] The elicitation of Non-functional requirements (NFRs) can be labor-intensive and time-consuming. Some researchers approach the problem of NFR extraction by using the Natural Language Processing (NLP) techniques based on user manuals and data agreements [57] and textual requirement specifications [43, 56]. Approaches that are NLP-based often assume that requirements documentation is clearly articulated. Unfortunately, NFRs tend to be poorly expressed in the requirements documents [29, 73, 4], and many software products being used in the real-world lack sufficient requirements documentation due to inadequate documentation techniques. This leads to difficulty in mining accurate and useful NFRs from textual content in practice. Furthermore, according to a recent study by Eckhardt et al., a classification of 530 NFRs extracted from 11 industrial requirements specification found that about 75% of so called "system-specific NFRs" were actually functional requirements [20].

In traditional requirements elicitation, researchers generally focus on requirements requested by stakeholders. User requirements collected in this way inevitably suffer from the issue of subjectivity, and the consequent software system may eventually fail for this reason [20]. By contrast, evaluation of system contexts and user behavioral data can be considered more objective, since this data can represent the communication between a user and system in real-world situations. Requirements engineers can identify NFRs through analysis of the user operations on the system and the corresponding system responses to the user. In this way, more complete requirements of users can be captured for system evolution.

Through human goal inference, recent work has successfully applied user behavioral and system contextual data to elicit functional requirements as task-level alternative features [67, 66, 6]. We believe that desires, as human mental states, are highly linked with and can be

projected to human goals. In this thesis, we treat human goals as a form of elaboration from desires, lower-level desires. Thus, it is possible to extract relations among desires at different levels of abstraction as NFRs based on contextual data.

Goal Oriented Requirements Engineering (GORE) considers NFRs as quality goals. Unlike functional goals, which are applied to create operational models (use cases and state machine models), quality goals select preferred options by comparing alternative ones and constrain on goal operationalizations [63]. Lamsweerde concluded that it is hard to establish a clear-cut sense for quality goals [63]. In a GORE process, requirements engineering activities are generally articulated by considering the intrinsical relationships among goals, agents, and scenarios [63]. This inevitably weak the human perspective and cannot consider NFRs well. However, we believe that the contributing relations among human desires have a better advantage to represent properties in NFRs since desires can only be derived from the human perspective.

The detection of desires from the human perspective has been studied and developed into a software development framework called Situ [12] where situation-awareness is of primary concern. A situation is defined as a 3-tuple: $< d, A, E >$ that consists of three contextual dimensions: the mental context (d), the behavioral context (A), and the environmental contexts (E). Mental contexts refer to human mental states which is largely hidden; behavioral contexts specify how humans feel out and influence the surrounding world; environmental contexts indicate the information that can be captured from the applications environment such as light, temperature, location, speed, etc. The Situ framework has been studied in the area of requirements engineering: requirements elicitation [66], intention identification [67], intrusion detection [65], and requirements prioritization [6]. These works also present the flexibility of adopting Situ in conjunction with a variety of models. We propose a multi-layered desire framework to analyze NFRs by focusing on new relations identified among desires at different layers based on the inference confidence of desires. The major difference between Situ framework and most other context-awareness research is that Situ explicitly includes human mental context. We use Situ framework [12, 10] as a model to build our methodology, since it supports elaboration on human

mental states (desires). NFRs are intrinsically subjective, relative, abstract [8, 52], interactive [14], and diverse [23]. Human desires have these characteristics in common, therefore, we will show a new method to map them into NFRs. Moreover, a situation sequence with time ordered observations is used to show the ways users pursue their interests within a system. As a result, we use the Situ framework to identify NFRs.

Previous work on Situ framework focused on requirements elicitation using Conditional Random Fields (CRFs), and through Multi-strategy, Task-adaptive Learning (MTL) method and the Strategic Rationale (SR) model. In this work, we present the following advantages to these approaches: 1) we infer of human desires from the inference of goals; 2) desires are treated as multi-layered; 3) we provide a quantified guideline to assist domain experts in analyzing and reasoning about users' new needs. We present a case study to illustrate our approach.

Our semi-automatic method can be used to improve workflow efficiency and engineer productivity during the process of NFR elicitation. With the Situ based NFR analysis, we made further progress to support system evolution at run-time, as anticipated in the research of Situation Analytics [10].

The contributions of this thesis are:

- Apply a CRF-based model to infer user desires at different levels of abstraction from the observation of user behavior and environmental contexts.
- Based on the inference confidence level, three methods are proposed to elicit NFRs by identifying new desires and contributing relations between them.
- Illustrate our NFR elicitation method with a case study, and show how to apply it in a real-world-like system.
- Extend the i* framework and provide a way to represent the elicited NFRs as task-level alternative features that developers can start to code.

- Elaborate some promising implications of this work from the perspectives of requirement prioritization, system architectures, system design, individualized requirement analysis, Agile development, etc.

The rest of this thesis is structured as follows. In Chapter 2, we describe related work and some background information for requirements engineering, NFRs, Situ framework, situ-awareness, sequential learning models, and i* framework. In Chapter 3, we compare concepts of desire and goal and discuss the relation between NFRs and desires. We introduce our method of desire inference and NFRs elicitation and present a running example to assist the interpretation of the method described in Chapter 3. In Chapter 4 we present a case study to illustrate the method and answer the following two research questions: (1) Can a CRF model be used to accurately infer humans' desires with different abstraction levels based on contextual data? (2) Can NFRs be elicited based on confidence level of inferring users' desires? We discuss the results of the case study in Chapter 5. In Chapter 6, we introduce an extension of i* framework, which provides a new link between human and system actors. In Chapter 7 we conclude by discussing some of the limitations of this work, as well as its potential in other research areas.

CHAPTER 2. BACKGROUND AND RELATED WORK

2.1 Requirements Engineering

Requirements engineering can be considered as a process of discovering, documenting, and managing the requirements for computer-based systems [58]. The purpose of requirements engineering is to generate a set of system requirements which is consistent, complete, and relevant and reflects what a customer actually needs [58]. The traditional process of requirements elicitation is an interactive learning process between the requirements engineers and the customers [41]. Through feedback, interview, and observation of customer activities, requirements engineers can obtain the knowledge of user requirements [33]. This process mainly relies on the subjective judgement of requirements engineers, which may lead to inaccurate requirements. After deploying a software system, it takes a long time to complete a software evolution cycle (elicitation of requirements, modification of design, development and deployment of the software system) to satisfy users' emerging needs [12]. Therefore, it is imperative for researchers to study systematic approaches based on engineering principles leading to better requirements and shorten the requirement elicitation process.

2.2 Non-functional Requirements

Non-Functional Requirements (NFRs) elicitation is a crucial component in the early phase of the requirement analysis of a software system. Approximately 50-60% of the errors found in software development are caused by incorrect requirements [51]. It results in increasing the cost of correcting errors in the later phases of the software development process by up to 100 times more than doing so in an earlier phase of the process [51]. Moreover, missing one NFR could possibly cause missing many users' needs. Thus, it is important to capture as many NFRs as possible in the early-phase of requirements analysis.

As a study on the taxonomy of software NFRs, Gazi et al. singled out usability, security, and performance as the most important NFRs of software systems such as information systems, real time systems, and web based systems [26]. In Afreen et al.'s taxonomy of NFRs, they classify usability, performance, reliability, security, and maintainability as commonly used NFRs [2]. Understanding and analyzing these NFRs well can help with better design decisions [75], requirement changes [32], and architectural evolution [61, 59].

NFRs focus on addressing the interests of stakeholders based on identifying a set of alternatives in system design [70]. Generally, NFRs tend to be stationary in a domain specific system. However, the NFRs of different stakeholders can be different or even in conflicted, and the same user may have different NFRs in different situations. This indicates that it is necessary to figure out users' NFRs in specific situations at runtime so that a better service could be provided.

Many mainstream methods for NFRs elicitation rely on techniques based on text mining and Natural Language Processing (NLP) [15, 74, 43, 56]. Those works proposed to apply various requirement documents to extract NFRs. They posit that different kinds of NFRs could be identified by considering keywords as label indicators. Those keywords are used to train an NFR classifier and require manual work to be identified from the documentation. An evaluation of 30 student software projects indicated that the results suffered a high ratio of false-positive [15]. Slankas et al. [57] proposed an automated method to extract NFRs from several kinds of software documents including requirement specifications, data agreements, user manuals, etc. Sentences in those documents were represented by word vectors and classified into 14 categories. They applied machine learning models such as K-NN, Naive Bayes, and Sequential Minimum Optimizer (SMO) to classify each individual sentence, and the results showed that SMO outperformed others in classifying requirement sentences. Zhang et al. [74] analyzed the performance of Support Vector Machines (SVM) with a linear kernel classifier using different textual structures to classify NFRs. Various types of NFRs were represented with three textual structures including multi-word phrases, individual words, and N-grams. They found that with Boolean weights, individual words had a better advantage to detect and classify NFRs, and a lot of NFRs were required to improve

the accuracy of inference. Casamayor et al. [9] proposed a semi-supervised method to detect and classify NFRs. After being trained with a small number of manually classified requirements, the classifier was then used to assign labels to unlabeled requirements using Expectation Maximization (EM). To further enhance the accuracy, users' feedback on classified requirements was exploited as well. Through an empirical evaluation, the proposed semi-supervised method showed better accuracy than learning methods that were fully supervised. These NLP based approaches assumed that NFRs exist in the documentation; however NFRs were not often documented, and even when documented, the documentations were not always precise and usually desynchronized. Therefore, the NFRs mined from documentations with NLP techniques tend to be inaccurate and can hardly represent user genuine needs from human-centered perspective.

One work that studied NFRs from the developer-centered perspective was conducted by Zou et al. [76]. They considered posts and discussions from the online Questions and Answers platform, Stack Overflow[1], as the text input, and a topic model was used to mine NFRs from developers' discussions. However, the limitation of their method was that end users' needs could not be easily reflected by developer-centric discussion. It takes an unpredictably long time period to prepare sufficient posts for analysis when collecting information from such online discussion forums. Moreover, the real meaning from the text of discussion cannot be revealed with a topic-model based technique. A possible way to overcome the limitations in studies would be to monitor system users' constantly changing behaviors, environmental context information, and mental states. This could provide a better advantage to obtain the NFRs in real-time from the perspective of users.

Some researchers focused on using machine learning and data mining techniques to prioritize NFRs. A representative work conducted by Duan et al.[19] for NFR prioritization applied clustering techniques to generate a list of prioritized requirements from stakeholder requests. However, their method requires human analysts to prioritize clusters and to weigh the importance of the clustering methods, which involve non-trivial human labor. Moreover, the prioritized

[1]https://stackoverflow.com/

requirements generated by their method cannot truly represent end users' real needs because the input to their method is a set of requests from stakeholders.

2.3 Situ Framework

In computer literature, there are various definitions of situation:

- "A situation can be an atomic situation, a logical composite situation or a temporal situation."

  —— Yau et al. [68]

- "The perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the future."

  —— Mica Endsley [21]

- "A triplet of a user's activities, a set of devices' actions and contexts over a period of time."

  —— Duckki Lee and Sumi Helal [39]

- "A situation at time t, Situation(t), is a triple $\{d, A, E\}_t$ in which d is the predicted user's desire, A is a set of the user's actions to achieve a goal which d corresponds to, and E is a set of environmental context values with respect to a subset of the context variables at time t."

  —— Chang et al. [12]

By considering situation as a 3-tuple: $< d, A, E >$ that includes human mental states (d), users' behavioral context (A), and environmental contexts (E), Chang et al.'s definition offers all the necessary elements to completely characterize a situation and provides a good abstraction containing the necessary components to compute a situation from the viewpoint of computer science [12]. So, we use Chang et al.'s definition of situation in this thesis.

The Situ framework assumes that human desires can be represented as a hierarchical structure called a desire tree. A compound desire is considered as the root of a desire tree and can

be further decomposed into smaller subdesires [12]. Desires in the leaf of the desire tree are atomic and cannot be further decomposed [12]. In this thesis, desires in two neighboring levels of a desire tree are treated as high-level desires, whose level is closer to the root of the desire tree, and low-level desires, whose level is closer to the bottom of the desire tree, respectively. A high-level desire tends to be more abstract, and may consist of multiple low-level desires. A low-level desire tends to be more concrete and independent.

In contrast to desires, the concept of goal is widely used in methods such as Goal Oriented Requirements Engineering (GORE) [64]. Lamsweerde defined a goal as "a prescriptive statement of intent that the system should satisfy through the cooperation of its agents" [64]. These agents are interweaved and cannot be separated from one another [51, 17]. A goal tree can be used to identify the traceability links from high-level strategic objectives of a system to low-level technical requirements. The literature of GORE demonstrates how goals from human users are catered into the system under development and that they are inseparable from the system perspective when requirements are being elicited [51, 17, 64].

Both desires and goals can be formulated at different levels of abstraction. A low-level desire, indicating a user's expectation to achieve a particular outcome, could be to "share a work" in a research collaboration system; a high-level desire, consisting of multiple low-level desires, could be to "build reputation" in a research collaboration system. Similar to the hierarchical structure in a desire tree, a low-level goal, which focuses on technical concerns, could be to "block account after 5 wrong password entries" for a banking system; a high-level goal, which specifies strategic concerns, could be to "serve more customers" for an online shopping system. The desire tree in the Situ framework is similar to the goal tree in GORE. The major difference is that a goal tree is analyzed from the aspect of systems to capture user requirements, whereas a desire tree is analyzed from the aspect of humans to monitor human intentions [12]. The multi-layered desires based framework proposed in this thesis is built on the desires tree in the Situ framework because it is human-centered and considers human needs in a hierarchical structure.

2.4 Situation-Awareness

We compare context-awareness with situation-awareness to distinguish these two concepts. The major factor that makes situation-awareness different from most other context-awareness research is the hidden mental states, which is presented as "desire" in Chang's Situ framework. In context-awareness, contextual data play a dominant role. However, human perspectives cannot be fully explored because the contextual data is only a resource for people to rely on and make decisions from. Context-awareness fits humans into contextual data and treat data as the first-class citizen [11]. Situation-awareness, on the contrary, requires fitting contextual data to humans and treat the human as the first-class citizen [11]. In situation-awareness, after a software application has been deployed, the constantly updated data collected from users in real-time plays a crucial role in system evolution. It is very likely that two individuals in the same context, behavioral or environmental, have quite different mental states, thus, different situations. Our work studies NFRs from a human-centric perspective, so the human mental states cannot be overlooked because humans deal with situations instead of only contexts in their daily life.

The research community of software engineering has already explored software requirements with situation analysis. Xie et al. applied the situation framework to requirements elicitation [66]. They used a CRF model to build the relationships between users' observations and their matching goals. However, studying the goal from a system perspective cannot full explore human needs. Also, considering goals as a single-layer concept overlooked the information between different abstraction levels of a goal tree. To automate the process of eliciting new requirements in the form of human intention, recent research conducted by Yang et al. [67] proposed to apply a Multi-strategy Task-adpative Learning method (MTL) upon the inference result achieved by CRFs. However, their method highly relies on the accuracy of CRF learning and is not able to cope with noises very well. This is because results not accurately predicted by a CRF model had not been sufficiently studied in their work. Atukorala et al. tackled the problem of requirement elicitation and prioritization [6] based on the structure of situation-transition which described a directed, weighted graph representing transitions from one situation to another. However, the

proposed method highly relies on observational data so the risks associated with unobserved situations could increase the uncertainty of requirements evaluation [5].

2.5 Sequential Supervised Learning Models

In many classical classification algorithms, the training data is drawn from a joint distribution independently; however, in sequential data, nearby data are tightly correlated [18]. Therefore, an appropriate learning model is needed to capture and parse relations between different observations and the inferred labels in a sequence of observation. To deal with sequential data in machine learning, many supervised learning algorithms have been employed such as conditional random fields [35], sliding window methods [18], hidden Markov models [7], graph transformer networks [38], etc. Machine learning models like hidden Markov models and sliding window methods are not able to appropriately describe the correlations among neighboring observations and labels. By using flexible feature templates, CRFs have an advantage to output feature functions to be used for capturing relations between parts of the correlated sequential data.

In terms of accuracy for tagging sentences in part-of-speech (POS) problems [18], named entity recognition (NER) [49], and handwriting recognition [22], CRFs have been proven to outperform HMMs (Hidden Markov Models) [7] and MEMMs (Maximum Entropy Markov Models) [44]. CRFs have also been applied on pattern recognition and machine learning [35], biological sequence data [42, 54], and intrusion detection [28]. In comparison to HMMs, CRFs are more efficient in interpreting the relations among desires, behaviors, and system context values in the Situ framework since they are capable of overcoming HMMs' limitations by utilizing flexible feature functions instead of a probability matrix. The concrete reasons of us to apply CRFs are explained in Section 3.2.2. Some existing popular open-source tools of CRFs are CRF++[2] written in C++, CRFsuite[3] written in C, Pycrfsuite written in Python[4], MALLET[5] written in Java, etc.

[2]https://taku910.github.io/crfpp/
[3]http://www.chokkan.org/software/crfsuite/
[4]https://github.com/scrapinghub/python-crfsuite
[5]http://mallet.cs.umass.edu/

2.6 i* Framework

Focusing on the early-phase requirement engineering (RE), i^* framework [70] is one popular method for modeling NFRs. The i* framework was proposed to model and reason about organizational environments and their information systems [70]. It aims to analyze and model stakeholder interests and how they could be addressed by various system-and-environment alternatives [67]. This framework consists of two major modelling components: Strategic Dependency (SD) model and Strategic Rationale (SR) model. The Strategic Dependency (SD) model describes the dependency relationships among various actors under an organizational context; it shows external relationships among actors, but hiding the intentional constructions within each actor [70]. The Strategic Rationale (SR) model describes stakeholder concerns and interests, and how they might be addressed, or compromised, by various configurations of environments and systems; it provides a more detailed level of modelling by studying the inside of actors to model internal intentional relationships [70]. The i* framework builds on a knowledge representation model to information system development [45]. It has also been used in business process redesign and modeling [71] and in software process modelling [72]. The i* framework was approved as an international standard by International Telecommunication Union (ITU) in 2008. We use this framework to represent the elicited NFRs as tasks-level alternative features because it helps answer the question "how" in relation to Strategic Dependency (SD) model and Strategic Rationale (SR) model, which is exactly the important characteristic of the NFRs. However, desires were not considered in the i* framework, which makes it hard to explore the constantly changing human minds and the evolving domain knowledge. Curtis et al. [16] identified the sparse spread of domain knowledge and changing requirements as the most critical risk factors for software project success. To fully understand the evolving domain knowledge from human perspective, we made use of the capability of i* framework and extended the current model by introducing certain human desire elements.

CHAPTER 3. RESEARCH METHODOLOGY AND PROCEDURES FOR THE CASE STUDY

In this chapter, we present the research methodology and propose the procedures of conducting a case study to support the methodology.

3.1 Desire vs. Goal

In this section, we first differentiate between the concepts of goals and desires. In addition, we describe how to elicit and model NFRs based on various abstraction levels of desire.

3.1.1 Goals

As reviewed earlier, in goal-oriented requirements engineering (GORE), a goal is characterized by the cooperation between multiple agents such as system and stakeholders. In this study, we analyze requirements in exploring genuine human desires through the lens of their own mental states. In this regard, system perspectives are not the main concern as is the case for GORE.

3.1.2 Desires

We therefore need to clearly differentiate between goals and desires. In our view, goals and desires are meant to surface as different levels of concerns. First, desires will emerge based on a human's mental state at that time. Then, in order to satisfy human's desires, system goals will eventually need to be identified. There is still some vagueness between human goals versus system goals. Our treatment about human desires will help derive human goals which will come to existence at a lower-level of abstraction (i.e. more specific) than that of the initial set of human desires. For example, if someone desires to become healthy (i.e. a desire), he will set his personal (i.e. human) goals as: 1) stick with a regulated daily routine such as sleeping at 10pm and

awaking at 6am, 2) go to the gym three times per week with vital signs tracked, and 3) adopt a new diet free of animal protein. As you can see, there are now three human goals to satisfy one human desire to become healthy. It is the application developers' task to develop a set of system goals to help the person to achive his personal goals, such as a reminder system, a Bluetooth-based wearable devices to track and record vital signs, and a diet recommender. In this way of distinguishing, desires and goals are two different concepts, while human goals can be considered as an interface to bridge between human desires and the eventual system goals. This is similar to the stepwise refinement concept traditionally found in software engineering, but here we emphasize the transition from the human domain to the system domain in a desire-decomposition, goal-refinement process.

Note that desire is one kind of mental states, whereas desires are often derivatives of common sense, emotions, moods, and human cognition. Thus, the desire-oriented approach we take focuses primarily on human desires in relation to their mental states. Studying NFRs from the perspective of desires could produce better representations of human mental states because desires tend to capture what a user genuinely needs and wants when compared with, more specifically, goals without running into the risk of being biased by system perspectives.

Computing literature in situation-awareness shows how researchers have proposed to visually represent human desires using a tree-like structure similar to a multi-layered goal structure [12]. Just as other researchers decompose goals in their studies, we decompose human desires into a multiple hierarchical structure with different abstraction levels. The root of a desire tree is compound and can be decomposed into smaller subdesires, while desires at the leaf node of a desire tree are atomic and cannot be further decomposed. In this thesis, we define desires situated in between two neighboring levels of a desire tree as high-level desires and low-level desires as follows:

Definition 3.1.1. *High-level desires are desires in the level that is closer to the root node of a desire tree.*

Definition 3.1.2. *Low-level desires are desires in the level that is closer to the leaf node of a desire tree.*

A Low-level desire can indicate a system user's eagerness for achieving a specific and desirable outcome; a high-level desire could consist of multiple low-level desires.

Although it is difficult to capture desires, the rapidly growing areas such as psychophysiology, affective computing, and brain informatics may enable software engineers to deal with human desire during the development and maintenance of software systems [10]. Furthermore, once desires are linked to actual actions, software engineers could be able to track them by making full use of the existing techniques including eye-tracking, gesture-recognition, and electroencephalogram devices in modern sensor-laden environments [10].

3.1.3 NFRs in Desires

We noticed that multiple ways exist in realizing a high-level desire with low-level desires. These alternative ways are indicative of quality constrains and can give a hint to NFRs from human perspectives. In order to achieve a high-level desire a user will first need to realize a low-level desire. A user can realize the a low-level desire towards achieving the high-level desire in two different ways: the way to realize the low-level desire helps achieving the high-level desire, in which case the low-level desire has a positive contribution to the high-level desire; or, the way to realize the low-level desire prevents from achieving the high-level desire, in which case the low-level desire has a negative contribution to the high-level desire. In our multi-layered desire framework, although both positive and negative contributions are used to identify users' evolving NFRs, the contribution links from low-level to high-level desires, used by domain experts to build the initial understanding of users' needs, only acknowledge positive contributions. This is because the positive contributions provide a clear indication of human mental states with the assumption that humans are rational. In this thesis, NFRs are viewed as desire contribution relations between desires in different abstraction levels since the desire contribution relations indicate how users want to address their interests.

3.2 Methodology

3.2.1 NFRs Elicitation Using the CRF Method

This section presents the methodology for eliciting new NFRs based on observation sequences using a statistical model CRF. The method of NFR elicitation that we use requires two steps: (1) inferring desires by using the CRF models, and (2) eliciting NFRs through the analysis of relationships between desires at different abstraction levels.

To clearly interpret the process of desires inference, we hereby declare a few concepts in the Situation-Aware research domain as follows:

- Context: information that surrounds the system, including visible effects due to environmental state change (environmental context) or human behavior (behavioral context), or invisible effects owing to the hidden human mental state (mental context). Upon detecting and perceiving the environmental contexts, certain human behaviors may be triggered pursuant to the human mental state at the moment.
- Action: manifestation of human behavior such as operation on a system via system interface.
- Situation: a snapshot of all the domain statuses at a time point, specifically time-stamped users' desires and actions and systems' contextual values.
- Observation: system status observed at a time instant including a set of time-stamped user actions and the corresponding system contextual values.

In this work, the CRF model was used as the computational model for desires inference and NFRs elicitation. We used CRF++, one implementation of CRFs with C++, because its accuracy has been proved to be better than other implementation algorithms of CRFs such as CRFsuite, MALLET, Wapiti[1], sgd[2], etc [48].

Before desire inference, a set of feature templates are defined to encode the relations between desires and observations, and to describe features used in the training and testing process.

[1] https://wapiti.limsi.fr/
[2] http://leon.bottou.org/projects/sgd

CRF++ requires the input data to have a fixed number of columns. Table 3.1 presents an example of the input data satisfying this requirement, where each line represents a situation, with the action in the first column, followed by the context value in the second column, then followed by the user's desire in the last column. For simplicity as an example, we only use one context value, although in reality it can be a vector of many contextual values. Next, we use the input data in Table 3.1 to explain feature templates.

Table 3.1 Input data in CRF++ format (the line with "<<" indicates the current situation)

Action	**Context Value**	**Desire**
a_1	c_1	d_1
a_2	c_2	d_2
a_3	c_3	d_3 <<
a_4	c_4	d_4
a_5	c_5	d_5

Table 3.2 describes an example of feature template used for the input data presented in Table 3.1. As presented in Table 3.2, each line denotes a template and a corresponding expanded feature and output label from the input data in Table 3.1. The macro $\%x[row, col]$ denotes the information of a token (situation in our case) in the input data where the *row* indicates the relative row position based on the current focusing token (the line marked with "<<" in Table 3.1), and the *col* indicates the absolute column position. For example, $\%x[0, 0]$ denotes "a_3" which is an expanded feature in the format of a string. Also, two tokens can be connected together to present a transition between two observations, such as $U04 : \%x[-1, 0]/\%x[0, 0]$ in Table 3.2.

Table 3.2 Feature template (U: unigram; B: bigram)

Template	**Expanded feature**	**Output label**
$U01 : \%x[0, 0]$	a_3	d_3
$U02 : \%x[0, 1]$	c_3	d_3
$U03 : \%x[-1, 0]$	a_2	d_2
$U04 : \%x[-1, 0]/\%x[0, 0]$	a_2/a_3	d_3
$B01 : \%x[-1, 1]/\%x[0, 1]$	c_2/c_3	d_3, d_2

A CRF model has two types of feature templates: unigram and bigram. Unigram templates ($U01 - U04$ in Table 3.2) describe the relation between observations and the current desire, while the bigram templates ($B01$ in Table 3.2) specify the relation between observations and the combination of the current and previous desires. Based on feature templates, CRF++ generates a set of feature functions automatically. In Table 3.2, an example of feature function generated by unigram template $U04 : \%x[-1, 0]/\%x[0, 0]$ is:

$$f_u(O, d_n, n) = \begin{cases} 1 & \text{if } d_n = d_3 \\ & \quad O_{n-1} = a_2 \\ & \quad O_n = a_3 \\ 0 & \text{Otherwise} \end{cases}$$

This function means that if the current desire d_n is "d_3", and the current observation O_n is "a_3", and the previous observation O_{n-1} is "a_2", then the function will return 1; otherwise return 0.

In Table 3.2, an example of feature function generated by bigram template $B01 : \%x[-1, 1]/\%x[0, 1]$ is:

$$f_b(O, d_{n-1}, d_n, n) = \begin{cases} 1 & \text{if } d_n = d_3 \\ & \quad d_{n-1} = d_2 \\ & \quad O_{n-1} = c_2 \\ & \quad O_n = c_3 \\ 0 & \text{Otherwise} \end{cases}$$

The above function can be interpreted as: if the current desire d_n is "d_3", and the previous desire d_{n-1} is "d_2", and the current observation O_n is "c_3", and the previous observation O_{n-1} is "c_2", then the function will return 1; otherwise return 0.

Each feature function is associated with a corresponding weight representing its inference reliability. To infer the desire sequence of the input observation sequence, the CRF model calculates the score of every possible desire sequence by adding up the weighted feature functions over all data in the observation sequence, and these scores are then transformed into probabilities

between 0 and 1 using normalization and exponentiation. The desire sequence with the highest probability will be used as the inference result for the input observation sequence.

We note that each feature template has its unique meaning, standing for the relations between observations and desires. Requirements engineers should define the template themselves under their study scenarios for both unigram and bigram empirically to find the optimum solutions. We take the stand that requirements engineers should know well the domain to be tackled. Even if they are not domain experts, they should be capable of working with domain experts to define feature templates. In our work, we prefer the former, that is, requirements engineers working on elicitation of application requirements should be domain experts. Thus, it is not necessarily burdensome for requirements engineers to define feature templates as part of the requirements engineering (formalization) task.

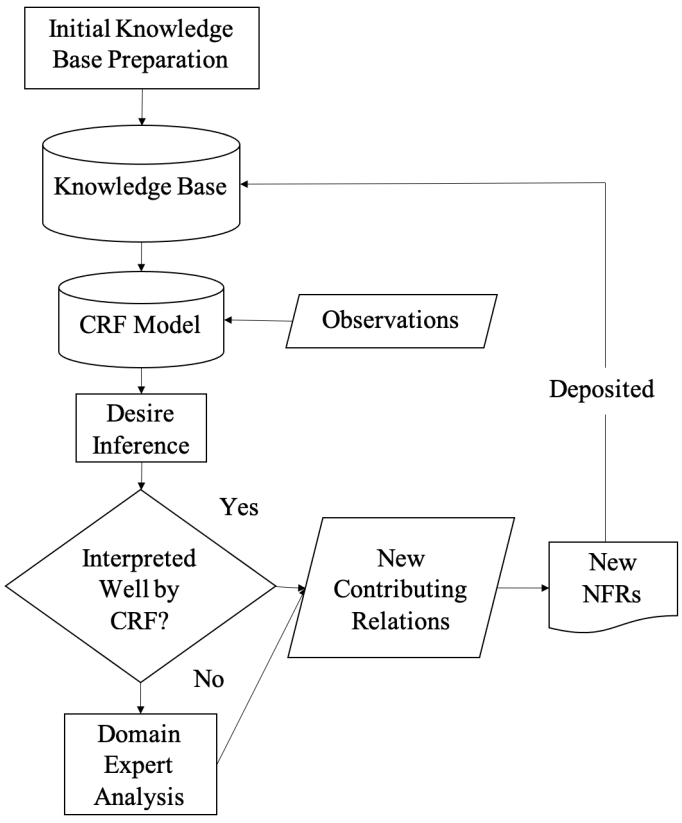

Figure 3.1 Workflow of NFR elicitation

The workflow of our method is presented in Figure 3.1. According to the workflow, the first step involves domain experts training a CRF model. This continues with the CRF model taking observation data as input and emitting inferred desires with confidence values as output. Using the inference confidence, experts can then discover new contributing relationships between the various levels of desires or decide which part of observation sequences should be further analyzed for new desires. Those newly identified contributing relationships and new desires will be employed for NFR elicitation. Finally, the elicitation results will be deposited into the ever-expanding knowledge base to be assimilated into the existing domain knowledge.

The detailed method is presented as follows:

1. Building the Initial Knowledge Base: As a supervised learning approach, the build of a CRF model relies on the existing domain knowledge. Generally, domain experts or proficient system users generate the input data to train the model. The format of training data is a set of combination of observations and desire sequences, such as $< o_1^*, (d_{a1}, d_{b1}, d_{c1}, ...)_1^* >, < o_2^*, (d_{a2}, d_{b2}, d_{c2}, ...)_2^* >, ..., < o_n^*, (d_{an}, d_{bn}, d_{cn}, ...)_n^* >$. The observation sequences $< o_1^*, o_2^*, ..., o_n^* >$, in which each observation (o_i^*) consists of users' actions and system environmental contexts (a_i, e_i), need to be collected from monitoring experts' or proficient users' behaviors, recorded by a monitoring mechanism in the system, when interacting with the system. The corresponding desire sequences $< (d_{a1}, d_{b1}, d_{c1}, ...)_1^*, (d_{a2}, d_{b2}, d_{c2}, ...)_2^*, ..., (d_{an}, d_{bn}, d_{cn}, ...)_n^* >$ should be either labeled by domain experts or self-reported by proficient users accurately. The sequence $(d_{ai}, d_{bi}, d_{ci}, ...)$ represents desires at different abstraction levels, with d_{ai} as the one with the lowest level.

2. Training the CRF Model: Requirements engineers define feature templates including both unigram and bigram feature templates based on analysis of the relations among users' desires, actions and context values [66]. Note that for each abstraction level of desires, a corresponding set of feature templates are defined independently. By feeding the training data to feature templates, a CRF model can be trained.

3. Data Collection and Pre-processing: Observations of user operations on the target system are collected. Specifically, user actions and system context values need to be recorded. The observation sequences,$< o_{11}, o_{12}, ..., o_{1j} >, < o_{21}, o_{22}, ..., o_{2k} > ...$, are pre-processed to the prescribed format.

4. Desires Inference: Once the CRF model and collected observations are ready, we use the model on the sequences of observation to label inferred desires, $(\hat{d_{ai}}, \hat{d_{bi}}, \hat{d_{ci}}, ...)$, for each observation. Similar to step 2, the processes of desire inference at different abstraction levels are independent from each other.

5. Situation Partition: For every two neighboring levels of desire, such as $(\hat{d_{ai}}, \hat{d_{bi}})^*$, from the lowest to the highest level in a desire hierarchy $(\hat{d_{ai}}, \hat{d_{bi}}, \hat{d_{ci}}, ...)^*$, set thresholds for inference confidence and separate inferred desires into relatively high confident and low confident parts based on these thresholds, and group situations into four cases, as presented in Table 3.3 based on their desires' inference confidence.

Table 3.3 Four cases used in our evaluation.

		High-level Desire	
		High Confidence	Low Confidence
Low-level Desire	High Confidence	*hh*	*lh*
	Low Confidence	*hl*	*ll*

6. Exploring Relations Between Desires at Two Different Abstraction Levels for New NFRs: For each case in Table 3.3, domain experts look for new contribution relations between neighboring levels of desires, such as $(\hat{d_{ai}}, \hat{d_{bi}})^*$, based on new desires identified from analyzing the situation sequences or based on existing desires with an automatic method. Those newly identified relationships imply users' emerging NFRs. Next, we elaborate each case for NFR elicitation, respectively.

High confidence high- and low-level desires (*hh*)

In this case, both low- and high-level desires are predicted with a high confidence, showing that both level desires are interpreted accurately by the CRF model and are very

likely existing in the initial knowledge base. The predicted low-level desires should be contributing to the high-level desire as we assume users are rational. Since each high-level desire might have multiple contributing low-level desires, a weight to each contribution link is computed based on the frequency of each contributing low-level desire. Among all the contributing relations between low- and high-level desires, if a contributing relation is not in the training set, it indicates an emerging contributing relation is found. We could add this newly identified desire contribution relation to the initial knowledge base as a complement if it has a specific value of weight. At the phase of building the initial knowledge base, domain experts might not take into account all the desire contribution relations between different abstraction levels of desires. Case hh in Table 3.3 indicates the promising opportunity to automatically identify and deposit new desire contribution relations into the initial knowledge base.

Low confidence high-level desires and high confidence low-level desires (lh)

Domain experts focus on the situations that have a low-level desire inferred with a high confidence, which is larger than a predefined threshold, and the corresponding high-level desire inferred with a low confidence, which is smaller than a domain specified threshold. Our explanation for this case is that the predicted low-level desire is included in the training data, but the predicted high-level desire is not. In other words, the low-level desire makes a positive contribution to some "unknown" high-level desire which has to be further examined by domain experts. To make such an "unknown" desire clear, domain experts should investigate neighboring situations for some useful hints. In practice, this newly identified high-level desire and its corresponding desire contributing relationship can be used to update the existing multi-layered desire-based framework when they are added to the initial knowledge base. The new high-level desire will interact with related existing desires in the training data. This new "desire" interaction could further impact the overall system design [67].

High confidence high-level desires and low confidence low-level desires (hl)

To handle the reversed circumstance of case (lh) as previous described, we focus on situations which have a high-level desire inferred with high confidence and a low-level desire inferred with a low confidence. This phenomenon could indicate that some other low-level desire, not included in the training set, may be contributing to the high-level desire. This new low-level desire means a new alternative for the high-level desire it is contributing to. The low-level desire is not interpreted well by the CRFs model primarily as a result of lack of specific knowledge in training set. To understand new low-level desires, domain experts refer to the surrounding situations to reason about. Through this point of view, new alternatives to realize existing high-level desires could be revealed.

Low confidence high- and low-level desires (ll)

In case ll, both low- and high-level desires are predicted with low confidence, smaller than the predefined confidence thresholds. We believe that users' desires (both low- and high-levels) are not in the initial knowledge base. Among all four cases, this is the most difficult case for domain experts to analyze. In the previous three cases, we have at least one level desire inferred with high confidence, which can be used as a trustworthy resource for further reasoning. However, for case ll as presented in Table 3.3, both level desires are inferred with ambiguous results, and there are a lot of possibilities for the potential low- and high-level desires. Thus, domain experts need to consider all the available information as comprehensively as possible for elicitation.

3.2.2 Why CRFs

A previous study investigated ways of applying a CRF model for goal inference [66]. We believe that CRFs could be a well-suited model for desire inference for the following two reasons: (1) Both goals and desires can be presented with tree hierarchies, and users' low-level desires are generally very concrete, and thus can be substantially projected to high-level system goals; (2) goals can act as requirements to help users to achieve their desires [12], so both goals and

low-level desires contribute to high-level desires. Based on this viewpoint, CRF appears to be a promising model to infer low-level desires.

In addition to underpinning low-level desires, we believe that CRFs can also be used to infer high-level desires, which are at a more abstract level. There are several reasons for this: (1) High- and low-level desires are very similar from the point of view that they all correspond to humans' mental states and can be reflected by the external behavioral sequences within specific environments; (2) the process of a user achieving a high-level desire from realizing a few low-level desires shows how a user achieves a particular high-level desire from several actions.

Similar to low-level desires, we can apply the same method to infer high-level desires by considering observations such as users' behaviors and environmental contexts. An alternative way to infer high-level desires is to consider not only the observations, but also inferred low-level desires as input data. However, the low-level desires inferred may include noise, to avoid feeding the CRFs model the noisy input, in this thesis, we only use observations as the input data of the model to infer high- and low-level desires.

We emphasize that the amount of effort in applying CRFs will be reasonable due to two reasons. The first reason is that training data, which is required for any supervised learning model, can be recorded by an embedded monitoring mechanism, such as computer vision, regardless of the specific of software applications. The second reason is that there can only be one of three ways to define feature templates: unigram, bigram, or a combination of both. In most cases, several consecutive observations can be considered for defining templates. This allows templates to be reused across systems conveniently within the same problem domain. Thus, we argue that using and reusing CRFs in observing, monitoring and controlling features of real-life software applications would not be difficult.

3.2.3 Some Statistical Methods Used in Data Analysis

In this section, we introduce some statistical method used for analyzing study results.

Mann-Whitney U test is a statistical test for analyzing if two independent samples come from the same distribution. This test is a nonparametric method without assuming data following any particular distributional form. MannWhitney U test prescribes the null hypothesis (H_0) that the two independent groups have the same distribution and are homogeneous. The test is based on the comparison of every datum from the one group with every datum from the other group. The null hypothesis is supported in the case that two groups come from the same population, as specified in the null hypothesis, where every observation of the first group will have an equal chance of being smaller or larger than every observation of the second group. The null hypothesis is rejected if the observations of one group is significantly larger or smaller than the observations of other group. General, the level of statistical significance, p-value, is used to decide whether to reject the null hypothesis or not. The p-value ranges from 0 to 1, which represents the probability that the observed difference between two groups is by accident. The smaller the p-value is, the stronger the evidence indicates to reject the null hypothesis. A small p-value ($<= 0.05$) indicates strong evidence against the null hypothesis, the null hypothesis can be rejected. A large p-value (> 0.05) indicates weak evidence against the null hypothesis, the null hypothesis cannot be rejected.

While a p-value shows whether an effect exists or not, it will not show the magnitude of the effect. Effect size (substantive significance) refers to whether an observed effect is large enough to be meaningful in practice, and it measures the magnitude of the difference between groups. Therefore, the workability of a treatment or intervention can be revealed by a significant p-value, whereas an effect size shows how much it works. One way to calculate the effect size is to consider the standardized mean difference between two groups: subtract the mean of one group from the mean of another group, then divide the result by the standard deviation of the population where two groups are sampled from. The effect sizes can be divided into different levels based on Cohen's classification: small (0.2), medium (0.5), and large (0.8) [60]. It is the best practice of researchers to interpret and report both the substantive significance (effect size) and statistical significance (p-value) in a quantitative study.

3.3 A Running Example

We hereby use a running example to illustrate the concepts introduced above and provide a detail explanation of the methodology and the procedures in a case study using the running example in Chapter 4. To explain the concepts and methodologies in our framework in a clearer way, we refer to Cooperative Research Environment (CoRE) [66], which is a real-world-like software system. The CoRE system was modified from "MyReview"[3], which is an open-source manuscript management system. The modification retained its basic functionalities and added necessary instrumentation code to record system context values, user behaviors, and user desires. We developed CoRE as an IRB-approved project meant for use by researchers as a platform to share their ideas and opinions on academic papers, as well as to collaborate.

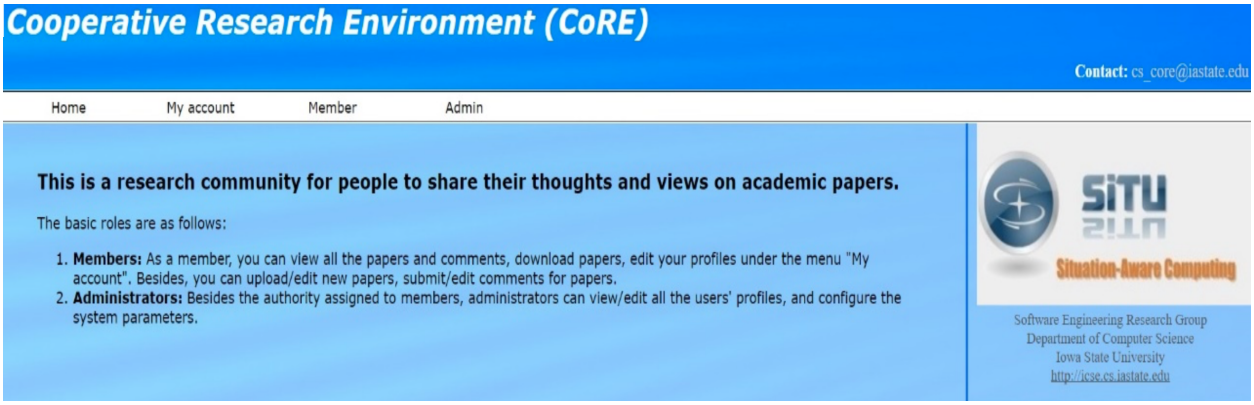

Figure 3.2 Main page of CoRE system

Downloading, uploading, and viewing papers, contributing their respective comments, as well as editing user profiles are some of the operations users can perform on this system. In addition to uploading and editing a paper, users can either submit and/or edit comments about a paper. Figure 3.2 shows the main page of the CoRE system. To login the CoRE system, a user can access the login page by clicking the link "log in" under the menu "My account". Users can access their profile page by clicking the link "My profile" under the menu "My account". On the profile

[3]https://github.com/ISCPIF/myreview

page, there is a form showing users' profile, including the login ID, roles, number of papers a user uploaded, and number of comments a user submitted. To view the list of papers and comments a user has submitted, the user can access the page "list of all papers" by clicking the link "View papers and comments" under the menu "My account". On the page "list of all papers", there is a form showing all the uploaded papers. Each row of the form shows one paper's information and operational links associated with the paper. Figure 3.3 indicates where users can upload a paper through the system interface. A user can submit a paper by providing information such as the paper title, key words, authors' names, etc., and uploading its electronic version as a PDF file. Additional details about CoRE can be found in Appendix . Our analysis of desires and NFRs under the CoRE system domain is based on some of the operations mentioned above. The availability of a system (could be a prototype) for data collection, and that there is some domain knowledge (maybe incomplete) and training data available are the prerequisites for our research.

Based on the CoRE system, we provide some examples for desires, desire hierarchy, as well as the representation of NFRs using desire contributing relations. In the CoRE system, an example of which human desire could be when a user indicates, "I want to discuss a topic with someone." This indication would instead be reflected as the system perspective: "The system should allow users to submit comments." The desire hierarchy is illustrated in Figure 3.4 with the high-level desire, "Find a collaborator." This particular high-level desire can then be separated into two separate lower-level desires, "Discuss" and "Share a work." The relationships between these multi-layered desires can then show how users want to indicate their interests. These relations are viewed as implicit NFRs. For example, to achieve the desire "Find a collaborator," a user could first share a work and then have a discussion with someone in the system. To elaborate the relations between desires in different abstraction levels, Figure 3.5 illustrates that to achieve the high-level desire "build reputation," a user could first need to realize the low-level desire "share a work." A user could realize the desire "build reputation" through "share a work" two different ways: either sharing a work in as much detail as possible or sharing a work as conveniently as possible. In the former circumstance, a user could provide as much material as

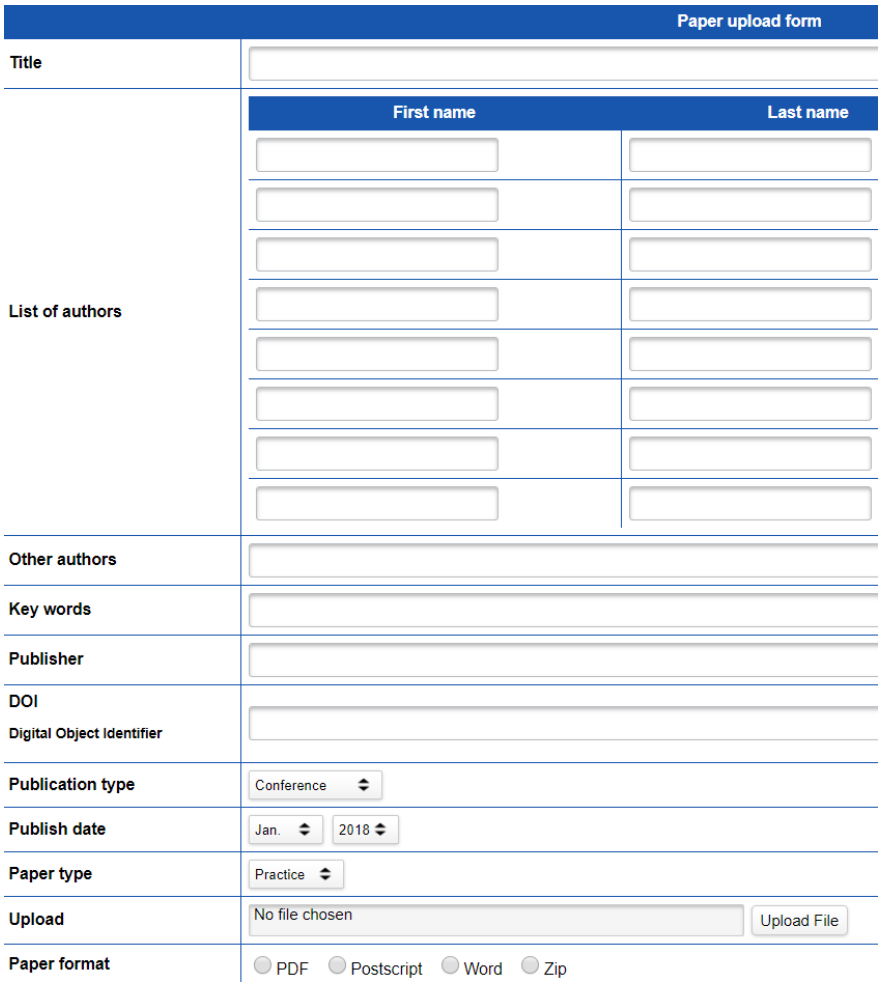

Figure 3.3 Upload a paper

possible when uploading a paper in the system. That makes a positive contribution towards achieving the high-level desire since it increases the volume of information of a paper. Alternatively, a user could prefer convenience and skip some information when uploading a paper, which makes a negative contribution to realize the high-level desire since it decreases the volume of information of a paper.

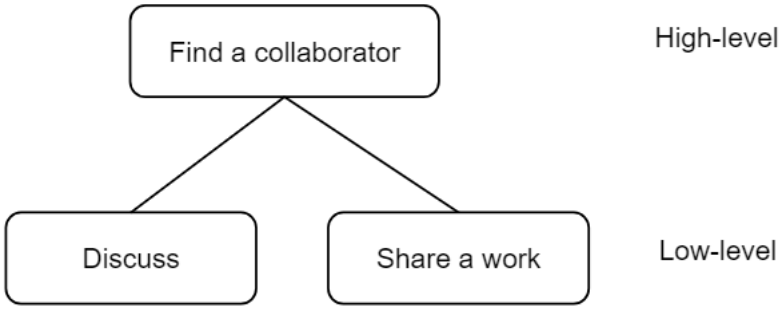

Figure 3.4 Desire decomposition

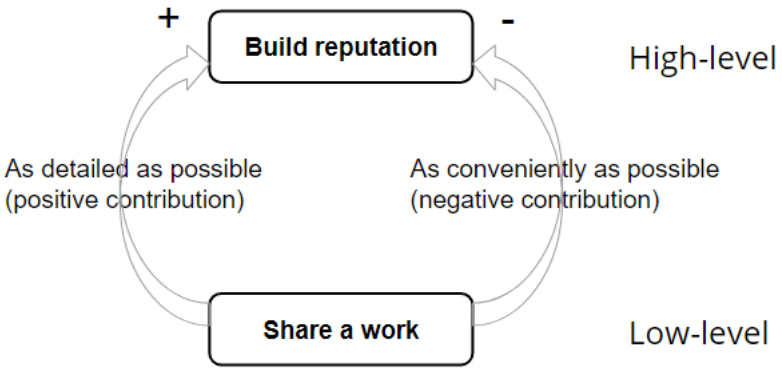

Figure 3.5 NFRs among desires

CHAPTER 4. A CASE STUDY

This chapter illustrates how our methodology using the CoRE system can be applied. To simplify our discussion, only two levels of desires are considered: high- and low-levels. We designed and ran an IRB (Institutional Review Board) approved case study to answer the following two research questions:

1. Can a CRF model be used to accurately infer humans' desires with different abstraction levels based on contextual data?
2. Can NFRs be elicited based on confidence level of inferring users' desires?

4.1 Study Design

Participants were asked to read the CoRE user manual before logging into the study, which included the instructions of operations in the system. The concrete steps a participant were instructed take in our study are described below:

1. Log in to the study using a pre-prepared link.
2. Complete a Pre-Session questionnaire to provide some demographic information, the familiarity with the CoRE system, and personal evaluation of the importance of RE.
3. Start a session by logging into the CoRE system.
4. Report each desire through two drop-down menus on the web page before moving to new desires.
5. Perform any operations, including, but not limited to, activities such as clicking buttons, clicking links, submitting inputs, browsing the web page, etc.

6. End a session to finish his operations on CoRE.

7. Take a Post-Session questionnaire page asking for feedback on that session.

For the purpose of increasing operations variety, participants were encouraged to choose a number of different activities to operate on the system, and they could operate on the CoRE for as many sessions as they wanted.

During each session, the following information was recorded for data analysis: users' login ID, users' actions (including mouse click on a button or a link, and selection on a drop-down menu), the time point when they performed an action, the current web page where the action occured, the next page that the system was directed to, contents on the web page (users' input information, system's responses to their action) and their self-reported desires. In our study, domain experts defined low- and high-level desire lists based on their expertise and experience with the CoRE system. Low-level desires included "share a work", "view a work", "revise my shared work", "save a work to local", "discuss", "withdraw my opinion, "locate my interested work , "view my shared work", "view my discussion", and "not in the list." High-level desires included "build reputation", "encourage somebody", "look for feedback", "look for collaborator", and "not in the list." Participants were given the flexibility to spend any amount of time they wanted to operate on the CoRE system.

4.2 Participants

We recruited participants from the departments of computer science from three different universities including Iowa State University in the United States, Northeastern University in China, and Nihon University in Japan. Table 4.1 summarizes the characteristics of participants. We collected this demographic information from a pre-session questionnaire, including gender, age, computer science background, familiarity with CoRE system, importance of software requirements, education level, and native language.

The total number of participants for our case study was 147. Among all the participants, 78% had a computer science related background. To measure their familiarity with the CoRE system, or their opinion about the importance of software requirements, we used a 7-point likert scale with 1 being "Not at all familiar" or "Not at all important" and 7 being "Extremely familiar" and "Extremely important". The average familiarity of participants was about 3.3, which is slightly familiar, and they tended to treat requirements as an extremely important (6.8) factor during the software development process. About 85% of the study participants held at least a bachelor's degree. The top three native languages of the participants were Chinese, Japanese, and English, respectively, comprising 80% of all the kinds of languages.

Table 4.1 Participant Characteristics

Participants	Number or percentage
Male/Female/Unknown	107/40/0
Average age	30.0
CS degree	78%
CoRE familiarity	3.3
Importance of requirements	6.8
Education level	
-Graduate degree (Masters, Doctorate, etc.)	46%
-Bachelors degree	39%
-Some college, no degree	14%
-Some high school	1%
Native language (top three)	
-Chinese	38%
-Japanese	24%
-English	18%

4.3 Data Collection and Preprocessing

The duration of this study was 6 months, from March to September 2018. In our case study, the data had been collected before the CRF model was built, since domain experts were also participants, and their records were separated out for later training purposes. For the software

applications with enough historical data records, it was completely fine to train the CRF model with historical data before collecting new data.

We collected a total of 11,960 raw records (or situations), corresponding to 949 sessions. A session included a sequence of situation records starting from a user's login to logout. Since the users' self-reported desires served as the ground truth for validating desires inference accuracy, it was critical to filter noisy records when participants did not perform conscientiously. Afshan et al. and Fry et al. tried to treat the completeness as a criterion for potential responses with high quality when they conducted crowdsourced software engineering studies [25, 24, 3]. Larlus et al. used Gold standard questions to assure participants payed continuous attentions during a survey [36, 31]. We follow similar rules to remove data noise, as described as follows:

1. We removed the sessions that were missing pre- or post- session questionnaires with the assumption that participants did not provide a good faith of effort.

2. In the post-session questionnaire, we asked participants if they had updated their desires every time when they had a new desire. If the answer was "No", the session was removed.

3. Based on the filtering results from the above two criteria, the number of sessions contributed by participants ranged from 1 to 62. Domain experts manually filtered those obviously low-quality records. One example was to check whether a participant had contributed substantial operations and reported desires.

By filtering out noisy data records, the final data set used for analysis included 4,388 records and 792 sessions.

4.4 Ground Truth

To help verify the inference results of the CRF model, we created an instrumentation by asking participants to report their desires when using the CoRE system. It was designed to help participants record their desires when using the CoRE features, through activating two drop down

menus as shown on the right side in Figure 4.1. Participants' self-reported desires served as the ground truth to test inference accuracy.

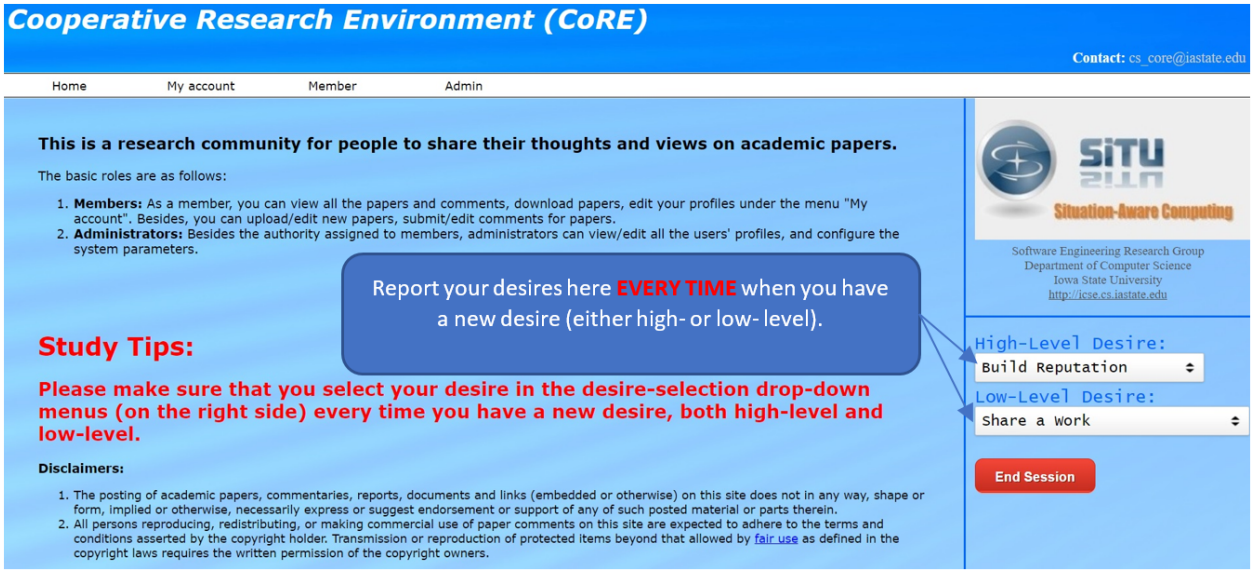

Figure 4.1 Interface for reporting desires in CoRE system

4.5 Procedures of Case Study

4.5.1 Initial Knowledge Base Construction

To obtain the training data, we separated data records generated by domain experts, including 318 sessions and 1,716 records. Table 4.2 shows the input training data, organized into three distinct columns. The first column is called Time Points, which show the specific time when situations occurred. The second column shows the situation sequence to achieve the high-level desire "Look for collaborator," while the third column shows another situation sequence to achieve another high-level desire, "Build reputation."

In Table 4.2, we note that an observation and a desire pair such as $< O_i, (D_l, D_h)_i >$ are the components of the situation S_i. Furthermore, each observation O_i includes expert action and related system context information, which is separated from the action by "&". For instance, the

Table 4.2 Training data examples: two situation sequences (d_{h1}: look for collaborator, d_{h2}: build reputation, d_{l1}: discuss, and d_{l2}: share a work)

Time point	$< O_1^*, (D_l, D_h)_1^* >$	$< O_2^*, (D_l, D_h)_2^* >$	...
1	(clickMenuAllPapers & (interval: 2s), (d_{l1}, d_{h1}))	(clickMenuUploadPaper & (interval: 3s), (d_{l2}, d_{h2}))	
2	(clickFilter & (interval: 6s), (d_{l1}, d_{h1}))	(clickComment & (interval: 111s), (d_{l2}, d_{h2}))	
3	(clickComment & (interval: 3s), (d_{l1}, d_{h1}))	(clickSubmit & (success; interval: 331s), (d_{l2}, d_{h2}))	
4	(clickSubmit & (success; interval: 87s), (d_{l1}, d_{h1}))	(clickMenuUploadPaper & (interval: 4s), (d_{l2}, d_{h2}))	
5	(clickMenuUploadPaper & (interval: 2s), (d_{l2}, d_{h1}))	(clickComment & (interval: 107s), (d_{l2}, d_{h2}))	
6	(clickComment & (interval: 75s), (d_{l2}, d_{h1}))	(clickSubmit & (success; interval: 252s), (d_{l2}, d_{h2}))	
7	(clickSubmit & (success; interval: 131s), (d_{l2}, d_{h1}))		
...	...	...	

situation at time point 4 in the second column shows that the "Submit" button was clicked by an expert, and the response from the system was successful. This situation also presents that the duration of the expert's operation was 87 seconds, and the high- and low-level desires of this expert were "look for collaborator" and "discuss", respectively. To have a better reflection of the time a user spends on a page, we only considered the operation duration between two consecutive actions as the system context information when preparing the training input. To fit the input format of the CRF++, we discretized the time interval into several levels as presented in Table 4.3.

Table 4.3 Discretized time interval of users' operation

Time interval level	Time duration (t)
T_1	0s $<$t $\leq 5s$
T_2	5s $<$t $\leq 10s$
T_3	10s $<$t $\leq 20s$
T_4	20s $<$t $\leq 30s$
T_5	30s $<$t $\leq$1min
T_6	1min $<$t $\leq$1hour
T_7	t $>$1hour

4.5.2 Feature Functions

By using CRF++, feature templates can be defined to generate feature functions for the training purpose. To train CRF models with CRF++, we applied both unigram and bigram to define the feature templates [66] for low-level desire and high-level desire inferences, respectively.

Tables 4.4 and 4.5 show the feature templates that we applied to generate feature functions for the CRF model. As described in Table 4.2, the input data of CRF include three columns: actions, time intervals, and desires. To infer low-level desires, we defined the feature template to capture the relations among several consecutive situations regarding the actions and time intervals because low-level desires tend to be concrete and may be reflected by a concrete action. For high-level desire inference, the feature template was formed to present the relations among the time intervals of several sequential situations since high-level desires are more abstract, whose feature template may not involve as many detailed actions as low-level desires do.

Table 4.4 Feature template for low-level desires

Unigram	Bigram
U01:%x[0,0]	B01:%x[0,0]
U02:%x[-1,0]/%x[0,0]	B02:%x[0,0]/%x[1,0]
U03:%x[0,0]/%x[1,0]	B03:%x[-1,0]/%x[0,0]
U04:%x[-2,0]/%x[-1,0]/%x[0,0]	B04:%x[-1,0]/%x[0,0]/%x[1,0]
U05:%x[-1,0]/%x[0,0]/%x[1,0]	
U06:%x[0,0]/%x[1,0]/%x[2,0]	
U07:%x[0,1]	
U08:%x[-1,1]/%x[0,1]	
U09:%x[0,1]/%x[1,1]	
U10:%x[-2,1]/%x[-1,1]/%x[0,1]	
U11:%x[-1,1]/%x[0,1]/%x[1,1]	
U12:%x[0,1]/%x[1,1]/%x[2,1]	
U13:%x[0,0]/%x[-1,1]	
U14:%x[0,0]/%x[1,1]	

Table 4.5 Feature template for high-level desires

Unigram	Bigram
U01:%x[-2,1]	B01:%x[0,1]
U02:%x[-1,1]	B02:%x[0,0]/%x[0,1]
U03:%x[0,1]	B03:%x[-1,0]/%x[0,1]
U04:%x[1,1]	B04:%x[-1,0]/%x[0,0]/%x[0,1]
U05:%x[2,1]	
U06:%x[-2,1]/%x[-1,1]	
U07:%x[-1,1]/%x[0,1]	
U08:%x[0,1]/%x[1,1]	
U09:%x[1,1]/%x[2,1]	

4.5.3 Desire Inference

We fed the training data and feature templates to CRF++ for model training. It took several iterations to train the CRF model before the improvement of the log likelihood between the current and the previous iterations was not larger than a threshold, which was set as a default value by CRFs tools.

Once the training of the models was completed, we pre-processed the collected data into the same format as the training data and fed them to the trained models for desire inference. The outputs of models were files that contained the original observation data followed by inferred desires, as well as the corresponding values indicating inference confidence.

Table 4.6 Inference result for input observations

Case	**Action**	**Time Interval**	**Inferred Desires (Confidence)**	
			Low-level	**High-level**
hh	A_{hh}	T_{hh}	$\mathrm{desire}_{hh}(\mathrm{P}_L > T_{Lh})$	$\mathrm{desire}_{hh}(\mathrm{P}_H > T_{Hh})$
lh	A_{lh}	T_{lh}	$\mathrm{desire}_{lh}(\mathrm{P}_L > T_{Lh})$	$\mathrm{desire}_{lh}(\mathrm{P}_H < T_{Hl})$
hl	A_{hl}	T_{hl}	$\mathrm{desire}_{hl}(\mathrm{P}_L < T_{Ll})$	$\mathrm{desire}_{hl}(\mathrm{P}_H > T_{Hh})$
ll	A_{ll}	T_{ll}	$\mathrm{desire}_{ll}(\mathrm{P}_L < T_{Ll})$	$\mathrm{desire}_{ll}(\mathrm{P}_H < T_{Hl})$

4.5.4 Situation Partition

We separated situations into different groups based on desire inference confidence. In our study, the inference results were divided into four combinations as presented in Table 3.3. Table 4.6 describes some examples of observations with desires predicted with different abstraction levels and different inference confidences. It also presents the situation partition based on the desire inference confidence with the first column being situation groups, followed by user actions in the second column, followed by the time interval in the third column, then followed by the inference results and confidences of low- and high-level desires in the last two columns. In Table 4.6, for low- and high-level desires, T_{Lh} and T_{Hh} were high confidence thresholds, and T_{Ll} and T_{Hl} were low confidence thresholds. Those thresholds were defined by domain experts based on the distribution of inference confidence level of desires. In practice, domain experts need to decide the confidence thresholds and to set the division for different abstraction levels of desires because these thresholds are generally domain specific.

4.5.5 NFRs Elicitation

For desires inferred with low confidence, the root cause could be divergent behaviors since these behaviors usually occurred when users operated in unexpected ways, which were not covered in the initial knowledge base built by domain experts. Considering situations containing divergent behaviors, it could be possible that users' actions were performed by mistake. However, if this kind of situations appear frequently, it is probably a hint for users' emerging new desires, which are not in the existing knowledge base. Thus, those situations with uninterpreted desires by CRF models can serve as a crucial resource for domain experts to conduct further analysis.

In Section 3.1.3, the NFRs were posed among desires. An NFR could be defined by an alternative that details how a user wants to address his or her interest, specifically how to achieve a higher-level desire through several lower-level desires. Therefore, after situations were separated into different groups, domain experts studied the relationships between desires with different abstraction levels for NFR analysis based on the method introduced at the end of Section 3.2. Due

to the vagueness introduced by the low inference confidence of both low- and high-level desires in casell, our discussion focused on the first three cases (hh, lh, and hl) of Table 3.3 in this thesis.

The CoRE system is a research environment to share opinions and collaborate with other researchers, and it has sufficient system handling capacity for the participants in the study. Therefore, among the most important three NFRs (usability, security, and performance) of software systems [26], the security and performance issues may not be well reflected on the CoRE system used in our study. Because of the characteristics of the CoRE system, the most obvious NFRs to be elicited in our study could be usability. However, our method are not limited to the usability study, and researchers could deal with other kinds of NFRs by incorporating corresponding training data in the initial knowledge base. Previous work had studied the intrusion detection from the security perspective with earlier version of the CoRE system [65].

CHAPTER 5. STUDY RESULTS

5.1 Desire Inference Results

Separated from training data, we had 474 sessions and 2,672 data records as test data. In our case study, if participants had a desire, either low- or high-level, that was not in the desire selection drop down list, they were given an option to choose "not in the list". Such records indicated that participants had some new desires that were not in the training data, so the CRF model was not able to interpret this data very well. These data could not be inferred correctly and had a negative impact on the desire inference accuracy. To explore the desires inference accuracy when users' desires were in the initial knowledge base, we filtered out those records when participants' self-reported desires were "not in the list." After filtering, the number of data records used to test low-level desires inference accuracy was 2,622, and records for high-level desires inference accuracy was 2,292. The accuracy of low and high-level desires was computed separately since their inference processes were independent. We computed the percent of correctly inferred desires as the overall accuracy for the test data, and the results showed that the accuracy of low- and high-level desires were 90.47% and 85.73%, respectively.

To understand the desires inference accuracy for different inference confidences, we divided desires into several zones by discretizing the inference confidence. Table 5.1 shows the desire inference accuracy as well as their percentage with different inference confidences for both low- and high-level desires. The accuracy of low-level desires tended to progressively increase with the increase of inference confidence, but this trend did not hold for high-level desires. There are two possible reasons: 1) the data collected were not enough to reflect a consistent variation as low-level desires; 2) high-level desires were more abstract than low-level desires, so the inference accuracy may vary with the different inference confidence. Our study achieved the accuracy of 99.18% and 95.32% for low- and high-level desires, respectively, when the confidence was over

90%. It revealed that the more confident the inference was, the more accurate the prediction would be, specifically when the confidence was above 70%. This shows that the CFR is a promising model for human desires inference in human-centric situation-aware domains.

Table 5.1 Desires' Inference results with various confidence (L: low-level desires; H: high--level desires)

Inference Confidence	0 - 0.1	0.1 - 0.2	0.2 - 0.3	0.3 - 0.4	0.4 - 0.5	0.5 - 0.6	0.6 - 0.7	0.7 - 0.8	0.8 - 0.9	0.9 - 1
Percentage (L)	0.00%(0)	0.07%(2)	4.57%(122)	1.91%(51)	1.42%(38)	0.67%(18)	0.82%(22)	0.67%(18)	0.86%(23)	87.13%(2328)
Accuracy (L)	0.00%	0.00%	11.48%	25.49%	21.05%	16.67%	22.73%	38.89%	56.52%	99.18%
Percentage(H)	0.00%(0)	0.04%(1)	0.11%(3)	0.67%(18)	1.76%(47)	6.77%(181)	4.42%(118)	4.19%(112)	11.23%(300)	58.42%(1561)
Accuracy (H)	0.00%	0.00%	33.33%	22.22%	12.77%	69.06%	64.41%	41.07%	81.00%	95.32%

One factor that might have impacted the inference accuracy could be the familiarity with the CoRE system. To understand the effect of CoRE familiarity, we divided data records into three groups based on participants' answers in the pre-session questionnaire. Data records were grouped as *familiar* if the answers in the questionnaire were "moderately familiar", "very familiar", or "extremely familiar." The *unfamiliar* group included records with responses including "not at all familiar", "low familiarity", or "slightly familiar." If the response was "neutral", then it was grouped as *neutral*. Table 5.2 presents the following results: for the low-level desires, the *familiar* group had a slightly higher accuracy than the *unfamiliar* group (93.76% vs. 91.19%), while for the high-level desires, the *familiar* group had a lower accuracy compared with *unfamiliar* group (76.68% vs. 82.21%). Overall, the accuracy of these two groups did not differ significantly across both level desires ($\alpha = 0.05$, Mann-Whitney U test) as revealed by the P values in the last column. One possible implication is that participants who were unfamiliar with the CoRE system were gaining familiarity while using the system, such that they were able to perform as well as those participants who were familiar with the CoRE system. Also, it indicates that the CoRE system is easy to use for users with various levels of experience.

Based on the above results, we answer the first research question: A CRF model can be used to infer humans' desire of different abstract levels accurately when the feature templates were defined appropriately.

Table 5.2 Accuracy comparison of records whose participants were familiar or unfamiliar with CoRE system

Desire levels	Familiar	Unfamiliar	Neutral	P-value
Low level	93.76%	91.19%	91.02%	0.0616
High level	78.68%	82.21%	83.29%	0.1459

5.2 NFRs Elicitation Results

As described in Section 4.5.5, NFRs were situated between low- and high-level desires, and we proposed to elicit NFRs based on the desires inference confidence of a CRF model. We assumed that users' emerging new desires might lead to the low inference confidence as they might have some new, divergent behaviors that could not be interpreted well by the CRF model. To verify this, we separated the records by whether the participants' self-reported desires existed or not in the training data and compared desires' inference confidence. Records were grouped into the *not existed* set if the reported desire of a record was "not in the list." Otherwise they were categorized as the *existed* set. The average desires' inference confidence was calculated for each set. Table 5.3 presents the results: for both low- and high-level desires, the inference confidence of the *not existed* set, when participants have new desires, was significantly ($\alpha = 0.001$, Mann-Whitney U test) lower than the *existed* set (low-level desires: 0.37 vs. 0.93; high-level desires: 0.57 vs. 0.88). The last column also shows the effect size (>0.8) was large based on Cohen's classification [60]. The substantial difference of inference confidence between the two sets indicated that the low inference confidence of a CRF model could be used as a hint to find new desires.

In the Post-Session questionnaire, participants were asked to report desire satisfaction degree for both low- and high-level desires using a 7-point Likert scale, with "1" representing "Completely dissatisfied" and "7" being "Completely satisfied". Table 5.3 presents the desire satisfaction degree for the two groups mentioned above categorized as *not existed* and *existed*. Similar to the inference confidence for both low- and high-level desires, the satisfaction degree was much lower in the set when participants had new desires than that in the set when they did not.

The significant difference was substantial as well. The low satisfaction degree of desires implies that users tended to be discontented with the CoRE system when they had new desires, which subsequently caused divergent behaviors and low desire inference confidence.

Table 5.3 Comparison of desires' inference confidence and desires' satisfaction degree between record sets with or without a new desire (L: low-level desires; H: high-level desires)

Desires' confidence/satisfaction degree	Existed	Not existed	P-value	Effect size
Confidence (L)	0.93	0.37	***<0.001	0.9420
Satisfaction degree (L)	6.40	4.50	***<0.001	0.8966
Confidence (H)	0.88	0.57	***<0.001	0.8975
Satisfaction degree (H)	6.00	1.70	***<0.001	0.9743

*α=0.1, **α=0.01, *** α=0.001

To analyze the different cases in Table 3.3 for NFRs elicitation, we set the thresholds of high and low confidence for the low- and high-level desires empirically referring to the distribution of inference confidence level of desires as presented in Table 5.1. Each threshold was assigned value as follows: $T_{Lh} = 0.9$, $T_{Hh} = 0.8$, $T_{Ll} = 0.5$, and $T_{Hl} = 0.5$. Also, researchers could try to learn the thresholds in a real-world scenario referring to methods such as reinforce method, bayesian optimization, and exhaustive optimization [40].

5.2.1 High confidence high- and low-level desires (hh)

To elicit NFRs automatically, we focused on desires of both low- and high-levels, inferred with a high confidence. We filtered the data records that had a predicted low-level desire with an inference confidence larger than T_{Lh} and had a predicted high-level desire with an inference confidence higher than T_{Hh}. Then, we looked for the contributing relations between low- and high-level desires for each high-level desire in filtered data set.

Table 5.4 details the contributing relations with the first column being high-level desires, followed by their contributing low-level desires in the second column, then followed by the contributing weights of each low-level desire in the third column. The last column shows the number of records and the corresponding sessions for each contributing relation. New

Table 5.4 Contributing relations and weights among high- and low-level desires for case hh ($*$ marks contributing relations not in the initial knowledge base)

High-level Desire	Low-level Desire	Contribution Weight	# Records (Sessions)
Build Reputation(30.07%)	Share a Work	0.582	311(66)
	Revise My Shared Work $*$	0.345	184(25)
	Discuss $*$	0.073	39(12)
Encourage Somebody(24.61%)	Discuss	0.641	280(59)
	Withdraw My Opinion $*$	0.336	147(40)
	Revise My Shared Work $*$	0.023	10(4)
Look for Collaborator(25.51%)	Discuss	0.587	266(57)
	Share a Work	0.413	187(54)
Look for Feedback(19.82%)	View My Shared Work	0.469	165(55)
	Revise My Shared Work $*$	0.375	132(33)
	Discuss $*$	0.128	45(20)
	Share a Work $*$	0.028	10(4)

contributing relations were marked with "$*$". Next, we elaborate on the contributing relations for each high-level desire.

Regarding the high-level desire "Build Reputation", which comprised about 30% of all the identified high-level desires, the most weighted subordinate and contributing low-level desire was "Share a Work" (0.582); however, for this high-level desire, we found another two contributing low-level desires, "Revise My Shared Work" and "Discuss" with weights of 0.345 and 0.073, respectively. Although the contributing weight of "Share a Work" was larger than the other two low-level desires, after discussing with the domain experts, we believe both these two new contributing relations were reasonable. The new contributing relations implicate that users could build their reputation via revising their shared work or discussing in addition to sharing a work in the CoRE system. Based on this, one potential improvement strategy of the CoRE system is to add mutual navigation links among pages on which those three low-level desires rely on. For example, on the page of "upload a paper", where users can realize a desire "Share a Work", to smooth the operability, users can either use a link or click on a Submit button that navigates them to the page of "edit a paper", where users can achieve a desire "Revise My Shared Work." This link or Submit button can be used in reverse.

For the high-level desire "Encourage Somebody", the two new contributing low-level desires identified were "Withdraw My Opinion" and "Revise My Shared Work". Regarding the low-level desire "Withdraw My Opinion", which had a non-trivial weight of 0.336, one possible reason is that users want to withdraw some comments that may discourage others. Based on the analysis, the other newly identified low-level desire "Revise My Shared Work", which had a relatively low weight, did not substantiate a contributing factor to the high-level desire "Encourage Somebody." To evolve the system and to increase the efficiency of operation, one possible modification to the CoRE system is to display a button or link for accessing users' comments history on the page where they want to submit comments for discussion.

In the case of the high-level desire "Look for Collaborator", there were no new contributing low-level desires found. However, the contributing weights of existing low-level desires indicate that users were more likely to realize the high-level desire "Look for Collaborator" through the low-level desire "Discuss" than via "Share a Work." This case can still provide useful guidance including prioritizing the potential mutual links among pages. For example, a link from the page of "upload a paper", where users can achieve a desire "Share a Work", to the page of "make a comment", where users can achieve a desire "Discuss", tended to be more preferred among users than the opposite case.

Regarding the high-level desire "Look for Feedback", we found three new contributing low-level desires. With the advice from domain experts, we have faith in all the contribution relations from the three low-level desires. Similar to previous cases, the CoRE system could be improved by adding transition links to allow users to jump between pages on which a user wanted to view or revise shared work.

Overall, in the case hh, we could detect new contributing relations among different levels of desires in the initial knowledge base based on their high inference confidence. In our study, the detected contributing relations could help domain experts develop improvement strategies for system usability such as operability and efficiency. All the newly identified contributing relations

could be used to update the initial knowledge base and to calibrate the CRF model for further system evolution.

5.2.2 Low confidence high-level desires and high confidence low-level desires (lh)

Case lh included data records that had a high-level desire inferred with a confidence lower than T_{Hl} and a low-level desire inferred with a confidence higher than T_{Lh}. For each data record that satisfied the pattern of case lh, we located all the records in the whole session to observe the neighboring behavioral and environmental contexts. To make sure that the filtered data records had reliable low-level desires, we limited the data set to keep only the sessions that had all low-level desires inferred with confidence larger than T_{Lh}.

As described in section 3.2.1, the high-level desires inferred with low confidence indicated users' potentially new high-level desires. In this case, our data showed that about 96% of data records had a unknown high-level desire reported. We present some examples to illustrate how to elicit NFRs based on identifying new high-level desires and new contributing relations from data records in case lh.

Table 5.5 Example of session in case lh (L: low-level desires; H: high-level desires)

Action	Time interval (s)	Inference result L (confidence)	Inference result H (confidence)
clickMenuUploadPaper	27	Share a Work (0.9814)	Look for Feedback (0.5041)
clickSubmit	23	Share a Work (0.9814)	Look for Feedback (0.3229)
clickSubmit	25	Share a Work (0.9612)	Look for Feedback (0.3183)
.....	...	.....	.....

One example we found in the data records was that a user tried to click the "Submit" button multiple times when uploading a paper. These data records corresponded to 6 (15%) sessions of 3 (19%) users in case lh. Table 5.5 lists one example of this phenomenon. It appears that the user was submitting a paper. However, clicking "Submit" multiple times meant their submission was not successful due to providing incomplete information as revealed from the observed data. Also, the related contextual data showed that the time interval for those operations of submission

tended to be short. We compared the time interval of submission operations between the records in this phenomenon and the records that had a users' self-reported low-level desire as "Share a Work" and high-level desire as "Build Reputation" because the contributing relation in the latter case had the highest weight, as shown in Table 5.4. The results showed that the average time interval of the former was much shorter than the latter (25.13s vs. 81.44s). The difference was statistically significant based on the Mann-Whitney U test ($\alpha = 0.001$). Such a phenomenon implies a user's preference of providing system required information as briefly as possible for convenience on the page of "Upload a paper" as illustrated in Figure 3.3. Therefore, we identified a new alternative of low-level desire "Share a work as conveniently as possible." Indeed, the desire "Share a Work" in the initial knowledge base means to share a work as detailed as possible, which makes a positive contribution to the high-level desire "Build Reputation."

To consider the "unknown" high-level desire, one potential explanation is that the users intended to perform a "quick update." If this is the case, the users might have intended to simplify the tedious process of submitting all information by submitting only some primary information when uploading a paper. As illustrated by Figure 5.1, the low-level desire "Share a Work" makes a positive contribution to the newly identified high-level desire "Quick Update" if a user chooses to share a work as conveniently as possible. The evolution of the CoRE system might focus on modifying the condition of submission by allowing incomplete information on the page of "Upload a Paper" to augment the system's operability and efficiency. Our elicitation is justified by participants' responses in the post-session questionnaire when the users were asked about the qualities they cared about when fulfilling the low-level desire "Share a Work". For the six sessions with this specific phenomenon, about 67% of the responses reported "flexibility" and 33% "efficiency" as qualities they cared about.

Another phenomenon we observed when analyzing data records in case *lh* is that 11 (69%) users submitted a comment consecutively, which was reflected in 21 (53%) sessions. As listed in Table 5.6, a user filtered a paper and submitted a comment, and then repeated those operations for a second time. In practice, the frequency of users' behavioral and contextual contexts can be

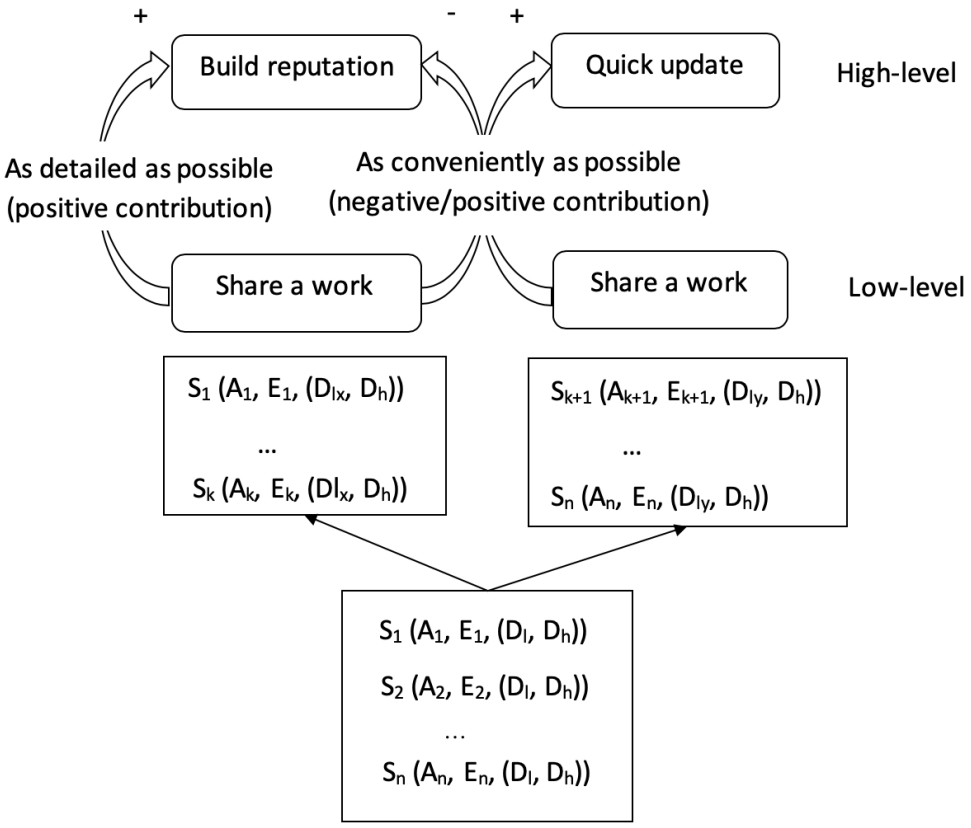

Figure 5.1 Discover new high-level desire (D_{lx} represents the desire to share a work as detailed as possible; D_{ly} represents the desire to share a work as conveniently as possible)

used as a hint to locate data records for analysis. For example, in this specific phenomenon presented in Table 5.6, those four repeatedly occurred users' actions ("clickMenuAllPapers","clickFilter","clickComment", and "clickSubmit") showed a frequency of 65% over all the actions in the data records in case lh. One possible explanation for the unknown high-level desire in this phenomenon is that the user intended to do a "batched comment" by submitting multiple comments one by one. Our analysis also gained support from participants based on their answers in the post-session questionnaire since participants mentioned "efficiency" in 95% of sessions and "flexibility" in 24% of sessions when asked about the qualities they care about. This newly analyzed high-level desire and contributing relation indicates that the

potential evolving strategies of the CoRE system could focus on providing interfaces to provide users with more operational efficiency from the perspective of usability. For example, they can submit batched comments more smoothly without having to jump back to the main menu to filter papers for each operation.

Table 5.6 Example of session in case lh (L: low-level desires; H: high-level desires)

Action	Time interval (s)	Inference result L (confidence)	Inference result H (confidence)
.....	...	.....	.....
clickMenuAllPapers	8	Discuss (0.9963)	Look for Collaborator (0.3699)
clickFilter	30	Discuss (0.9980)	Look for Collaborator (0.3939)
clickComment	23	Discuss (0.9963)	Look for Collaborator (0.4744)
clickSubmit	43	Discuss (0.9881)	Look for Collaborator (0.4389)
clickMenuAllPapers	11	Discuss (0.9947)	Look for Feedback (0.5324)
clickFilter	32	Discuss (0.9967)	Look for Feedback (0.4936)
clickComment	16	Discuss (0.9975)	Look for Feedback (0.4993)
clickSubmit	22	Discuss (0.9991)	Look for Feedback (0.4494)
.....	...	.....	.....

5.2.3 High confidence high-level desires and low confidence low-level desires (hl)

To analyze case hl, we separated out data records that had a low-level desire inferred with a confidence lower than T_{Ll} and a high-level desire inferred with a confidence higher than T_{Hh}. Similar to the filtering procedures used in case lh, we kept the whole sessions with all the high-level desires inferred with a high confidence larger than T_{Hh}. The low confidence low-level desire inferred by a CRFs model implicated an emerging new low-level desire, as mentioned in section 3.2.1. Our study showed that for data in the case hl, 89% of records were reported with an unknown low-level desire.

We apply an example to illustrate how to elicit NFRs in this case. In the process of reviewing those data records, we found that many users liked to click the button “MyProfile” on the main menu immediately after sharing a work. This phenomenon corresponds to 5 (22%) users and 12 (30%) sessions in case hl. Table 5.7 lists an example of such phenomenon. The first three records

Table 5.7 Example of session in case *hl* (L: low-level desires; H: high-level desires)

Action	Time interval (s)	Inference result L (confidence)	Inference result H (confidence)
clickMenuUploadPaper	2	Share a Work (0.9980)	Build Reputation (0.9625)
clickCommentPaper	105	Share a Work (0.9972)	Build Reputation (0.968)
clickSubmit	90	Share a Work (0.9649)	Build Reputation (0.9692)
clickMenuMyProfile	1	**Share a Work (0.4719)**	Build Reputation (0.9657)
.....	...	.....	.....

in Table 5.7 show that the user was trying to build a reputation via sharing a work due to the high confidence desires for both high- and low-levels. However, the last record of the low-level desires, shown in bold in Table 5.7, indicates an emerging new low-level desire since the inference confidence is lower than the predefined threshold. One potential interpretation of this new low-level desire is that the user wanted to check his or her profile since the user's behavioral context showed that the button "MyProfile" was clicked. Therefore, "Check My Profile" may be a new alternative way to build reputation.

The future improvement of the CoRE system could be adding a transition link to users' profile information on the confirmation page of "upload a paper" to augment the users' operational flexibility. To verify our elicitation, we checked the participants' responses in the post-session questionnaire. Sessions were reported with concerned qualities such as "flexibility" (58%) and "efficiency" (42%) when sharing a work, and we received the support from domain experts as well.

All three cases *hh*, *lh*, and *hl* illustrate how to elicit users' emerging NFRs such as efficiency, flexibility, operability, subjective satisfaction, etc. from the perspective of system usability. Therefore, we can answer Research Question 2 by stating that it is possible to elicit NFRs based on users' desire inference confidence.

5.3 Summary of Case Study

In summary, we provide three promising means to discover users' evolving NFRs based on desire inference confidence levels. For all the cases listed in Table 3.3, we focus on the first three

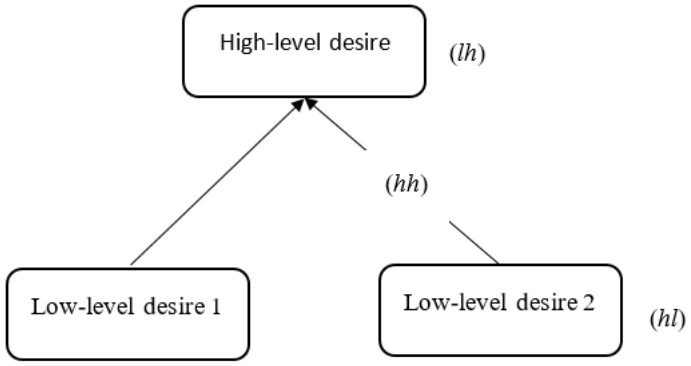

Figure 5.2 NFRs presented between two different abstraction levels of desires (lh, hh, and hl are mapped to the cases described in Table 3.3)

cases, illustrated by the desire tree in Figure 5.2. The case hh identifies new contributing relations between existing high- and low-level desires; case lh indicates the presence of a new high-level desire as well as the contributing relation from its corresponding subordinate low-level desires; case hl implies that a new alternative contributing to an existing higher-level desire is identified as well as the contribution relation between them.

To locate the objective situations for analysis, domain experts can consider both desires' inference confidence and appearance frequency of situations. Considering the frequency of situations, it may be difficult to locate situations that are exactly the same. However, objective situations are likely to be located by looking for specific patterns. For instance, in case hh, any situations with the identical high- and low-level desires can be used as the pattern; in cases lh and hl, the pattern may be any situation sequence that includes a certain percent of situations satisfying the conditions in each case, which can also be used to reduce the manual work load of domain experts. To look beyond the neighboring level desires in practical application scenarios, we suggest domain experts to explore the whole desire tree with a bottom-up fashion: reasoning from the lowest-level desires to the highest-level desires. The reason is that the revisions on the root (highest) level desire generally have a relatively large impact on all the subordinate level of

desires, while revisions on a bottom level desire may not have much impact on the root level desire or low-level desires in other branches.

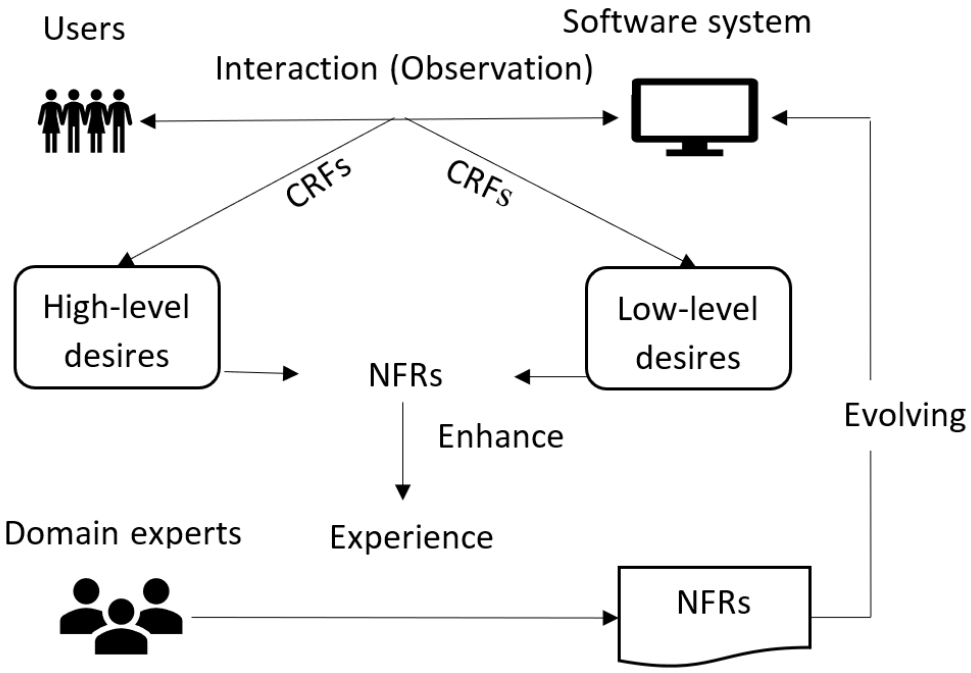

Figure 5.3 The outline of multi-layered desires oriented NFRs analysis

Based on a CRF statistical model, our NFR analysis focused on the contribution relations between desires by exploring users' genuine desires under the situation framework. We outline our method of NFRs analysis with Figure 5.3. The whole process starts from the interaction between users and the software system. Through this interaction, the users' behaviors and system responses information will be generated and serve as input for a CRF model to infer users' desires at both low- and high-levels. Since desires are generally subjective, we try to achieve the objectivity from observing users' behavioral and environmental contexts, instead of inferring desires directly. The relationships among different abstraction levels of desires could be presented based on the analytical procedures introduced in Section 3.2.1, such as the new desire contribution relationships (case hh), high-level desires (case lh), and alternative low-level desires (case hl). By analyzing the desire contribution relationships between desires, we can reveal the

information of how a user intends to address his or her interests. This information is a valuable resource for domain experts to elicit new NFRs from users' minds and figure out strategies of modification for the system-to-be.

CHAPTER 6. NFRS REPRESENTATION WITH AN EXTENDED i* FRAMEWORK – i*D

This thesis focuses on NFR elicitation from the human perspective, irrespective of the system itself, to ensure what the users get is what they want. To illustrate the difference between desires and goals, we distinguished these two concepts in Section 3.1. A desire only pertains to a human, while a goal is related to the interaction between a system and a human. Although desires and goals are concepts from two different perspectives, they are interrelated to some extent. Currently, the relations between desires and goals are still blurred in spite of decades of research [50, 62, 69]. Our work tries to bridge the gap between desires and goals in requirements engineering and to provide a formal way to show our elicitation results.

We have shown that we are able to elicit users' NFR requirements based on the new contributing relations and new high- and low-level desires from users' behavioral and contextual data. NFRs elicited by domain experts represent the relations among human desires and can be used by requirement engineers and software engineers for system evolution. However, there is still a gap between the analytical results of humans' desires from domain experts and the requirements that can be directly used by software engineers, which is a common topic in the research area of information systems [1, 53]. We take advantage of the ability of i* and extended the current model with human desire elements. Then, we present how to transfer our elicitation results provided by domain experts to system requirements that can be directly used by software engineers with the extended i* model.

6.1 i* Framework

The i* framework [69] provides a requirement modelling framework that supports the early phase of requirements engineering. It aims to analyze and model stakeholder interests, and how

those interests might be addressed by exploring alternate system proposals. The i* framework was approved by the International Telecommunication Union (ITU) as an international standard in 2008 [67]. This framework includes two modelling components: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. The former is used to represent the intentional dependency relations among actors, while the latter is used to represent the internal relationships between different intentional elements of actors, such as between stakeholder interests and concerns, and how they might be handled by different system and environment configurations.

6.2 i* Extension

Although the i* framework has tried to explicate issues and concerns that actors have about the current system with the SR model, the concepts used for descriptions of the human actors, such as hard goals, software goals, and tasks, are generally mixed with concepts used for descriptions of systems. To have the characteristics of a system and its users fully explored, we extend the i* framework as i*D (D signifies Desires) by separating the human actors from other agents in the system. In framework i*D, only desires with different abstraction levels are used in an SR model to describe the relationships between intentional elements inside each human actor. Figure 6.1 presents all the graphical notations used in the original i* framework as well as a new notation we introduced to represent desires in i*D framework. In this section, we illustrate how to use the extended SR and SD models to describe users' mental states and NFRs, respectively, while the elicitation results discussed in Section 5.2 are used as examples.

6.2.1 Extended Strategic Rational Model

Because the Strategic Rational model in the original i* framework does not have the element of desires, we add a new element to represent human desires in the i*D framework since the i* framework starts a process analysis from understanding the interests and motivations of system users. We only use desires as intentional elements of human actors to avoid interference from other systems agents by focusing on human mental states.

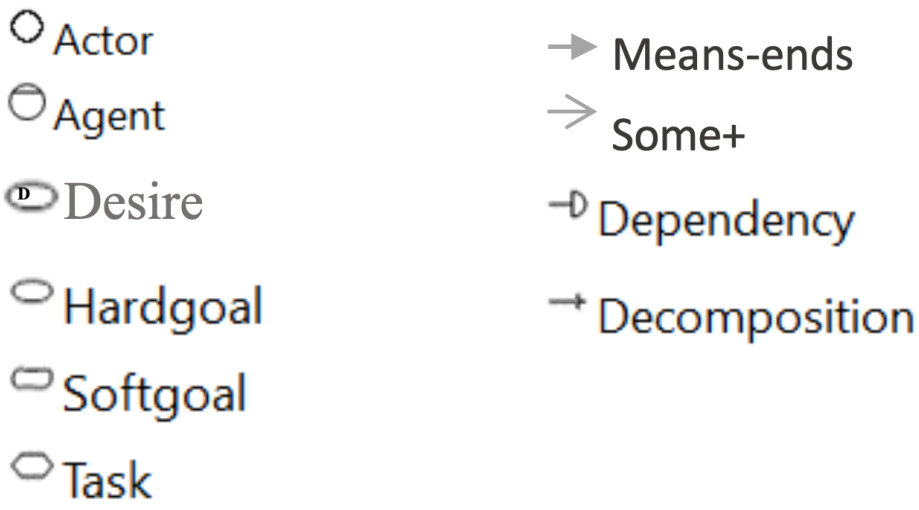

Figure 6.1 Graphical notations in i* and i*D framework

Our i*D framework also extends the formalization of the original i* framework because a more formal approach is able to offer the foundation for building tools to assist reasoning [27]. In the original i* framework, the analysis of SR model is discussed around four concepts including **ability**, **workability**, **viability**, and **believability** [69]. Our formalization of the extended SR model is described based on these four concepts. Since we treat human actors differently in an extended SR model, our description of formalization is focused on human actors. We use $\mathcal{D}_i$ to denote a desire at a certain abstraction level i and use R to denote a set of relationships among desires through which a user can address his or her interests. Desires with different abstraction levels are connected with means-end links since they are used to specify purposes and the ways to achieve the purposes, which is well matched to desire contributing relations. We describe the means-end link with a predicate $mel(l, \mathcal{D}, r)$ where l is the link, $\mathcal{D}$ is the end referring to an intentional element desire, and r is the means referring to a desire contributing relation (routine). The formal characterizations of ability, workability, viability, and believability are presented as follows.

Ability describes a human actor's process for realizing a desire. The following notation denotes that a human actor h has the ability (A) to realize a desire $\mathcal{D}$ if and only if the actor has a desire contribution routine r among all the routines (R) to do so. The *purpose* in the notation describes that the desire contribution routine r is specific for the achievement of the desire $\mathcal{D}$.

$$A(h, \mathcal{D}) \equiv \exists r(R_h(r) \wedge purpose(r, \mathcal{D}))$$

Workability describes whether the ability process is going to work or not. The desire $\mathcal{D}$ is workable for the human actor h ($W(h, \mathcal{D})$), if that actor has a workable desire contribution routine to achieve $\mathcal{D}$ ($W(h, r)$). To describe the workability of a desire contribution routine, we denote $element(\mathcal{D}, r)$ as all the desires in a desire contribution routine r.

$$W(h, r) \subset (R_h(r) \wedge \forall \mathcal{D}'(element(\mathcal{D}, r) \supset W(h, \mathcal{D}')) \wedge \forall r'(subroutine(r', r) \supset W(h, r')))$$

The workability of a desire contribution routine can be evaluated recursively. For example, a desire contribution routine is workable if all the desires in the routine are workable and all the subroutines of this routine are workable.

Viability describes how well the process to achieve a desire will work, which specifies the qualities of routines. Believability emphasizes on the evidence needed for a process to be workable. Viability and believability can be reflected by the workability of means-end links. We denote means-end rules of a human actor h as $\mathcal{H}_h(\mathcal{D}, r, \varsigma)$ where $\mathcal{D}$ refers to the desire, r refers to the means, and ς refers to the applicability situation. A workable means-end link (*mel*) l depends on the condition that the human actor has a rule for the desire contribution relations and the actor believes (B) the applicability situation ς of that rule to be valid, which is represented in the following notation:

$$W(h, l) \subset \exists \mathcal{D} \exists r(R_h(r) \wedge mel(l, \mathcal{D}, r) \supset \exists \varsigma(\mathcal{H}_h(\mathcal{D}, r, \varsigma) \wedge B(h, \varsigma)))$$

Figure 6.2 illustrates how to use the extended SR model to describe the new contributing relations between high- and low-level desires with one example for each case in *hh*, *lh*, and *hl* mentioned in Chapter 5. It can also describe the newly identified desires as annotated by

"unknown" in Figure 6.2. Contributing relations among different levels of desires are represented by means-end links, which are used in the original i* framework as well. For example, regarding the high-level desire "Build Reputation" (end), one alternative way (means) to realize is through achieving a low-level desire "Share a Work".

In the extended SR model, we associate a means-end link with its weight for desire contributing relations identified by case *hh* in Table 3.3, as *hh* is the case where both low- and high- level desires are inferred with high confidence. The desire contribution weight between two desires in different abstraction levels is defined as follows:

$$W(d_i, d_{i-1}) = \frac{|Situ_{d_i-1}(d_i)|}{|Situ(d_{i-1})|}$$

where d_i and d_{i-1} refer to two neighboring low- and high-level desires, respectively, $Situ_{d_i-1}(d_i)$ refers to the set of situations that have a desire d_i which contributes to the desire d_{i-1}, and $Situ(d_{i-1})$ refers to the set of situations that have a desire d_{i-1}. The desire contribution weight $W(d_i, d_{i-1})$ between neighboring desires d_i and d_{i-1} is calculated by dividing the size of set $Situ_{d_i-1}(d_i)$ by the size of set $Situ(d_{i-1})$.

As shown in Figure 6.2, for the high-level desire "Build Reputation", the weights associated with the means-end links from its three subordinate low-level desires are 0.58 ("Share a Work"), 0.35 ("Revise Shared Work"), and 0.07 ("Discuss"), respectively. This weighted means-end reasoning in the extended SR model provides quantified guidance to domain experts and requirements engineers for figuring out users' preferred system configurations. For example, in the CoRE domain, users are more likely to achieve the high-level desire "Build Reputation" through the low-level desire "Share a Work" compared with the low-level desire "Discuss" as shown in Figure 6.2.

6.2.2 Extended Strategic Dependency Model

By separating human actors from system actors, the i*D model brings new relations between these two groups of actors. The extended SD model specifies the dependency relations between human and system actors by adding dependency links to connect desire contributing relations

(depender) and the specific system tasks (dependee). For example, Figure 6.3 illustrates the dependency relations between the CoRE system and its users described by the SD model.

In Figure 6.3, the system actor "Papers and Comments Managing Module" (PCMM) is described with an SR model by connecting the system's hard-goals, soft-goals, and tasks with the means-end links and the task decomposition links. The dependency links in Figure 6.3 connect desire contributing relations with specific tasks in a module of the CoRE system. For example, the contributing relation between the high-level desire "Batch Comment" and the low-level desire "Discuss" depends on the implementation of a system task "Submit Comment for Multiple Papers." This system task has a positive contribution to the system soft-goal "High Usability" as shown in Figure 6.3. Thus, with the assistance of the extended SD model, user desires and system goals are connected by system tasks because creating a task for submitting multiple comments that bring users more operational efficiency can satisfy both the user desire and the system goal. The desire contributing relation mentioned in the above example implies users prefer some efficiency when submitting multiple comments. This can be served as an NFR that is mapped to the system goal as high usability.

We hereby provide a formal characterization for the relations between the human and system actors. The example shown in Figure 6.3 illustrates that a human actor relies on the elements that belong to another non-human actor, and the human actor has no knowledge about whether those elements are workable or not because he or she does not have the routine to achieve those elements. In the original i* framework, the commitment and transfer of workability have been used to bridge the gap of delegating workability among different actors [69]. We apply the same concepts to describe the dependency relations between human actors and non-human actors as follows:

$$W(h,t) \subset \exists s(A(s,t) \wedge C(s,h,t))$$

where an element t is workable for a human actor h if there is a non-human actor s who has the ability A to realize t and is committed (C) to realizing t for the depender h.

The NFRs in this thesis are measured within an individual's mental states, and in the CoRE system, there is not too much interaction among users. In practice, for an application where users frequently interact with each other, there may be some desire dependency relations among different kinds of users. For example, in a meeting scheduling system, the desire "attend a meeting" of meeting participants could depend on the desire "initiate a meeting" of a meeting initiator. In this case, the relation among different human actors can be described as follows:

$$W(h, \mathcal{D}) \subset \exists h'(A(h', \mathcal{D}) \wedge C(h', h, \mathcal{D}))$$

where a desire $\mathcal{D}$ is workable for a human actor h if there is a another human actor h' who has the ability to realize $\mathcal{D}$ and is committed to realizing $\mathcal{D}$ for the depender h.

6.2.3 Benefits of i*D Framework

In the i*D framework, the NFRs can be represented as an organic whole of desire contributing relations and their dependees (system tasks), which are connected by the dependency links. This is consistent with the concept of NFRs dictating "properties that functional requirements should have" [51]. In Figure 6.3, the task "Submit Comment for Multiple Papers" indicates a functional requirement, while its depender specifies a property with desire contributing relations that describe how users wanted to address their interests. Thus, users desires and system goals are indirectly connected by the dependency links in the i*D framework. The elicitation results from domain experts can be connected to the concrete tasks that can be directly used by developers with an i*D framework.

The i*D framework brings about a promising advantage to fully explore the evolving domain knowledge and promote software project success. With the i*D framework, the whole process of requirements elicitation, either functional or non-functional, will start from understanding human actors' interests and end up with practical system design tasks.

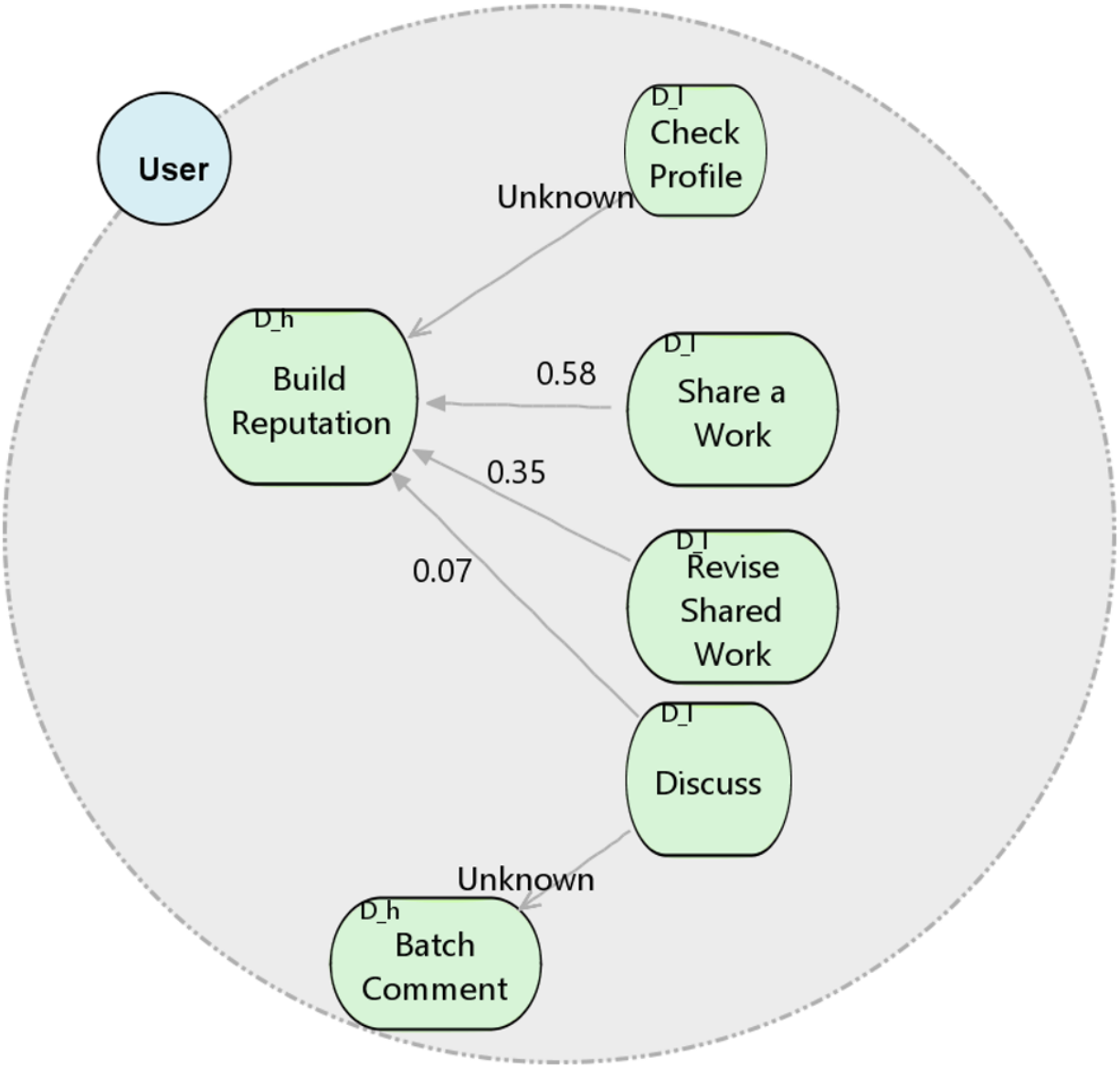

Figure 6.2 New contributing relations and new desires described by SR model in i*D framework (D_h: high-level desire; D_l: low-level desire)

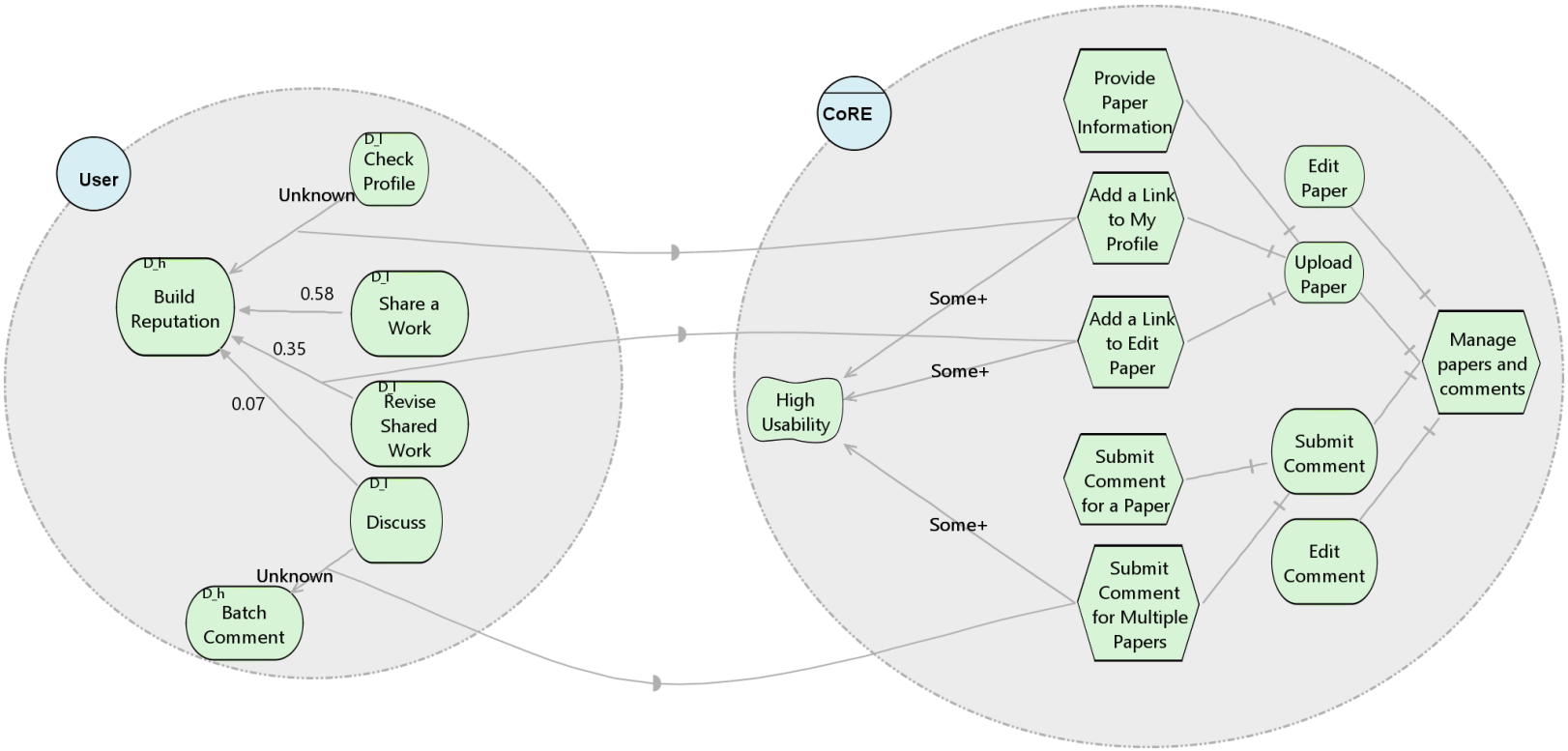

Figure 6.3 NFRs representation in CoRE for the system evolution with SD model in i*D framework (PCMM: Papers & Comments Managing Module)

CHAPTER 7. DISCUSSION AND CONCLUSION

7.1 Discussion

Other NFRs. The elicited NFRs from our case study basically focus on but are not limited to system usability. Considering the characteristics of the CoRE system with the current version, we do not expect many security issues because the system is used to provide researchers a friendly environment to share and collaborate which is not a mission-critical application. Due to the limited number of users in our study and the sufficient system handling capacity of the current CoRE system, we expect few performance issues to be reflected neither. However, in real-world applications, our framework can also be used to identify NFRs such as security and performance of a software system by incorporating corresponding training data in the initial knowledge base and by analyzing user observations such as login and page loading contexts based on inferred new desires or new contributing relations. For example, the case study by Xiao et al. has validated that it is possible to detect intrusion scenarios in a previous version of the CoRE system, which illustrates system issues from a security perspective [65].

NFRs prioritization. Prioritizing requirements is an imperative task for stakeholders since almost every software project is subjected to time to market constraints and budgetary restrictions [19]. Researchers has studied requirements prioritization under Situ framework based on situation-transition structure applied to elicit human-centered requirements [6]. Their method highly depends on observational data in a specific domain, and the unobserved situations and situation transitions will impact on finding the priorities of the derived requirements. Our method takes advantage of those unobserved situations to identify users' evolving needs and prioritizes requirements based on desire contributing weights. For example, case *hh* shown in Table 5.4 presents the prioritized contributing relations among different levels of desires with contributing

weights. The desire contributing weights could be used as a guidance for domain experts to make decisions about those most important NFRs when they are facing a tight deadline.

Mining NFRs from requirement documentation. Many methods [15, 74, 57, 9, 76] automatically extracting NFRs relying on techniques such as text mining and natural language processing (NLP). These methods share the assumption that various types of NRFs can be characterized by certain keywords, which are used as indicators. However, these methods basically suffer from the high false positive rate or low accuracy. Although these methods have strength in speed, they cannot well reflect the essential meaning of the requirements documentation. Our method tries to explain users' mental states by desire inference, and represents a very promising direction to combine the desires-oriented framework and NLP-based techniques. One example is to accelerate the construction process of the initial knowledge base in desires-oriented framework with automatically mined NFRs by NLP-based techniques.

Agile Development. An important incapability of Agile development is that it lacks convincing methods in dealing with non-functional requirements when coping with high-integrity software systems [13]. Our method hints an opportunity to improve Agile development from the aspect of requirements augmentation. In Agile development, the core characteristic is to advocate time-boxed iteration cycles with constant user engagement. Using our method, the NFRs elicited in each iteration could serve as task-level alternative features that will be considered in the next development iteration. Additionally, our method prefers an incrementally shippable system or an incremental prototype to be available, and a prototype can generally be constructed quickly when the Agile method is applied. Therefore, our method shows promise of being appropriate for Agile development considering its essence of rapid incremental and iterative refinement of the software products.

Moreover, because many low-fidelity prototypes aim to identify user needs, usability goals and the overall high-level conceptual ideas about a system [46], the desire hierarchy structure in our method can be used to assist building a system prototype from the perspective of understanding and validating user needs in early design. If a system prototype is absent, domain experts can

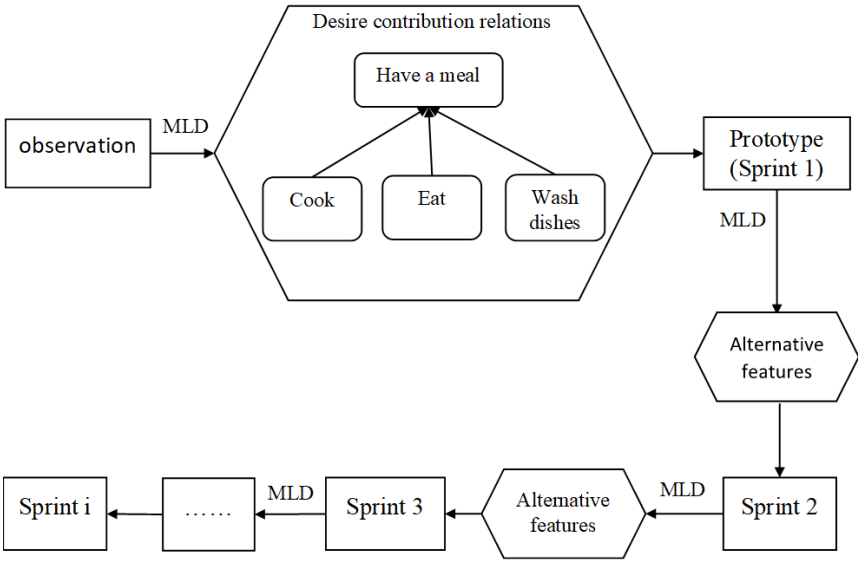

Figure 7.1 System evolution of a recognition system for activities of daily living with Agile development (MLD: Multi-layered Desires Based Framework)

figure out some initial assumptions about how a system-to-be is supposed to work by observing how a user functions without such a system. When they observed more of the users' behaviors and the corresponding effect in the environment, the initial observations can be useful for desire analysis and quickly constructing an early prototype. For example, Figure 7.1 illustrates the Agile development for a software system to detect users' activities of daily living (ADLs). To build a prototype, we can start from observing users' behaviors and environmental effects from existing history data, such as the pattern showing that users are likely to achieve a high-level desire "Have a meal" with several low-level desires including "Cooking," "Eating," and "Washing dishes." Those initial observations could help domain experts form an early-phase knowledge base and help develop a system prototype. Alternatively, if there exists a prototype, the application of our method can start from the Sprint 1 (prototype), as shown in Figure 7.1, and keep providing new features for the coming iterations.

Alternative ways for desire inference. In this work, we treat users' desires with different abstraction levels, and each level of desires can be inferred independently because they are human mental states that can be represented with external behaviors. Because Xie et al. [66] only dealt

with a single-level goal, the feature templates used for CRFs training need to be modified in our case. We note that multiple-level desires could have feature templates with different complexities. We believe that the lower level a desire is at, the more concrete it should be and the more specific feature templates are needed for successful recognition. For lower-level desire, their feature templates may be more sophisticated. However, for higher-level desires, in order to avoid the over-fitting problem, feature templates could be defined in a more general way to present the summary characteristics of their contributing lower-level desires.

Another way to predict users' higher-level desires is to use not only observation data, but also the inferred low-level desires. For instance, in Table 4.6, to infer the high-level desire located in the last column, their corresponding low-level desires could be taken into consideration. In this case, the input data of a CRF model can be either a combination of users' actions, environmental context values, and the inferred low-level desires, or just the inferred low-level desires. To determine what properties should be used in the combined input, the criteria need to be formulated empirically regarding the specific, practical problem. We note that the precondition to infer high-level desires with inferred low-level desires is that low-level desires are inferred with high confidence, which could be estimated by checking the inference confidence of CRFs models.

Hierarchical division of desires. With the goal of finding users' emerging intentions by analyzing their behaviors and system contexts, situation-aware computing has been used in requirements engineering. The most similar work to ours was conducted by Xie et al. [66], who utilized a CRF statistical model to infer users' goals based on the operations of users and system's corresponding contexts. Furthermore, they also proposed three analysis methods for eliciting users' emerging intentions by considering goal transition, users' divergent behavior, and erroneous behavior, respectively. Compared to their work, our approach has several advantages, four of which are especially notable. First, we take a further step in gaining a better representation of human mental states by using true desires inference instead of merely relying on goals inference. Second, the goals are treated as a single layer concept by Xie's et al., while we treat desires as a multi-layer concept by dividing them into multiple layers for analyzing their relations among each

other. Third, many records in the data pre-processing procedure were removed by Xie et al., while we take advantage of data records when user-reported desires do not exist in the desire selection dropdown list. Lastly, we suggest a more concrete metric of Section 3.2.1, which can be used as a means of reference for domain experts when analyzing new NFRs.

Human-computer interaction (HCI). The desires inference model provides us an opportunity to predict the user's desires during runtime. The inferred desires could assist software engineers to adjust requirement specifications and evolve software systems to meet users' satisfaction. After improving the system with users' genuine desires, users will continue to interact with the evolved software system, and the information generated from the interactions may provide insights for the next evolution. This process keeps iterating by continuously exploring human mental states and observations to help software services evolve better.

Also, the CRFs model consistently evolves as a result of users interacting with the system, leading to a better understanding of NFRs by software engineers. At the beginning, only domain experts are familiar with the software requirements, which is the only resource to generate the training data. After software is deployed, more and more observations coming from the field become available, which can be used to enrich the training data set and calibrate the model. Then, software engineers can recognize certain situation changes for further analysis.

Individualization. By exploring human mental states, our method shows a promising individualized software service paradigm. Individualization is superior to traditional personalization through parameterization as it considers individual user's specific requirements [34].

That is, the state-of-the-art personalization is mainly implemented with parameterization. In [30], several parameters in a collaborative filtering-based recommendation system were analyzed to generate user recommendations. With parameterization, researchers explore the combinations of different values for each parameter variable, within a specific value range, to generate different recommendations for different groups of users. Thus, this method tends to personalize the system for a group of users because it is impossible to exhaust all the possible combinations of parameter

variables. However, individualization considers the individual user's mental states, which are distinct for every user, to customize services for a single user. Admittedly, there is still a very long way from capturing users' complete mental states when using software, but this work presents a beginning of studying software requirements from extensively tackling human desires and it brings the insight of exploring human mental states into requirements engineering.

Runtime evolution. Unlike context-aware computing [55], which explores changes in context, such as system error logs, to discover users' intentions, the new methods incorporating human mental states bring the possibility of runtime individualized software evolution [10]. Assuming that we can accurately infer human mental states, it is still hard to implement and deliver the constantly evolved systems at runtime. We noticed that researchers in the software engineering community have applied genetic programming (GP) for automatic runtime bug repairs [37]. Further direction of runtime evolution might be using GP to generate service alternatives for users based on their mental states.

Although there are many challenges, such as recognition of users' favorable alternatives, runtime testing and system reliability, the information gathered from humans' mental states is extremely valuable for software engineers during the system runtime after the software has already been deployed.

Existing NFRs models. In the i* framework, software engineers and domain experts work together to construct models for analyzing the interests of various stakeholders and understanding how their interests could be addressed. The process of building this framework tends to be labor-intensive and subjective because communication between domain experts and stakeholders is usually through interviews, and it is hard to obtain the genuine requirements of the real users if they are not directly involved. Our method focuses on managing users' desires and the contribution relationships among them in order to support domain experts after software systems have been deployed. Domain experts might take advantage of the analysis results of our method to calibrate the i^* model to describe more accurate, individualized alternatives and actors' relationships with updated domain knowledge.

Usability study. The research community of user experience (UX) has applied both qualitative and quantitative analysis methods as their best practice, such as the diary study, field study, and evaluation methods [47]. These method have some limitations: 1) data collection may not be completed automatically, and subsequently the data need to be hand-coded; 2) participants either may not report all the information needed for analysis or act inconsistently when they are observed; and 3) it is unrealistic to conduct enough interviews with users to get a representative sample for analysis.

A part of the framework proposed in this thesis provides the mechanism for automatic collection of observation data, which overcomes the diary and field studies with regard to the reliability of collected data. Moreover, the desire analysis based on users' behaviors helps UX researchers understand whether users' desires are satisfied or not and further understand the usability of the system. As user experiences are subjective for each individual user, it could be advantageous to study the users' mental states before and after system deployment, instead of only the context information.

Functional Requirement Elicitation. Our framework could be able to elicit not only NFRs, but also functional requirements (FRs) through domain experts' examination. In our case studies, many NFRs elicited focus on system usability. However, in order to improve the usability, a certain amount of modification or new functionality should be made or added. For example, Table 5.6 shows that a user needs to jump between papers to give batched comments on the current system. To augment the operational efficiency, new functions, such as allowing a user to check batched papers and then to comment together, could be considered for the system-to-be. Thus, if the functionality does not exist in the current software system, it is possible that the observation of situation sequences could lead to new functional requirements.

7.2 Threats to Validity

7.2.1 Threats to Construct Validity

Desires reporting mechanism. To obtain the ground truth for verification of desires inference accuracy, we asked participants to report their desires while they operated on the system. Reporting desires is critical to our study; however, this process may affect participants' performance in the study. For example, a participant's seamless thinking of achieving a high-level desire may be interrupted by having to report several low-level desires. To understand the extent to which the desire reporting operation influences the process of fulfilling desires, we asked participants to respond with a 7-point Likert scale with 1 being "Not at all influential" and 7 being "Extremely influential." Analysis results show that the average influence value is 2.4, which maps to "Low influence" in our scale. Thus, the effect of reporting desires should not be a big concern.

7.2.2 Threats to Conclusion Validity

NFRs elicitation. The new NFRs elicitation based on desires inference confidence was conducted by domain experts by observing participants' situation sequences. It is still subject to question whether the elicited NFRs are really desired by the participants or they are just domain experts' subjective assessment. To validate this, we asked participants to indicate the operational qualities they cared about when conducting basic operations, such as uploading a paper and submitting a comment, on the CoRE system because this can largely reflect users' real needs. We found that the NFRs elicited by domain experts were also supported by participants' responses in the Post-Session questionnaire.

7.2.3 Threats to Internal Validity

Desires granularity. To make the CRF model produce reliable results, desires should be defined with a proper granularity. If the granularity of desires is defined too coarse, the CRF model may not be able to generalize the relationships between user desires and contextual data.

If the granularity of desires is defined too fine, the CRF model can be confused by labeling excessive tags. In our case study, it happens that the desires are relatively easy to define in the CoRE system domain, and in practice, domain experts should define the desires with appropriate granularity while considering concrete domains.

Data quality. Considering the quality of the data collected, it is possible to involve some undesired records if participants did not exert due effort in a good faith. For example, a participant might just randomly click through the whole study without paying any attention, and their data might lead to inaccuracy in desires inference and NFRs elicitation. Admittedly, this is a common problem for most evaluation studies that have human subjects involved. To mitigate this threat, we employed post-filtering and gold standard questions, mentioned in Section 4.3, as criteria to filter out the obviously undesired data.

7.2.4 Threats to External Validity

Domain. In the thesis, CoRE system was used as an example to explain the NFRs elicitation method. We strongly believe that our method will be applied in other human-centric context-aware domains since the training process may not be difficult to transfer across systems as mentioned in Section 3.2.2. The applicability of our method on other domains needs to be validated in the future.

7.3 Conclusion

This thesis proposes a method to analyze and elicit NFRs from contribution relations in human desires based on a situation framework. We distinguish desires with different abstraction levels and treat NFRs as how users achieve a high-level desire through different low-level desires. By focusing primarily on human-centric factors in goals, our primary contribution is to analyze NFRs from a human-centered perspective, which we believe is an appropriate fit and closely corresponds to the essential characteristics of NFRs. We also take inference confidence into consideration to make better sense of the inference results of CRFs models. We reason through a

case study and analyze three different cases, new contributing relations in desires, new high-level desires, and new low-level desires, respectively. Our results show the promise of our method in NFRs elicitation especially from the perspective of software system usability. We also extended from traditional GORE the broadly accepted i* framework to help incorporate NFRs elicited through using our method into the requirements analysis outcomes.

REFERENCES

[1] R. L. Ackoff. From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9, 1989.

[2] N. Afreen, A. Khatoon, and M. Sadiq. A taxonomy of softwares non-functional requirements. In *Proceedings of the second international conference on computer and communication technologies*, pages 47–53. Springer, 2016.

[3] S. Afshan, P. McMinn, and M. Stevenson. Evolving readable string test inputs using a natural language model to reduce human oracle cost. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pages 352–361. IEEE, 2013.

[4] D. Ameller, C. Ayala, J. Cabot, and X. Franch. How do software architects consider non-functional requirements: An exploratory study. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 41–50. IEEE, 2012.

[5] N. Atukorala. Situation-oriented requirements engineering. PhD thesis 2019.

[6] N. L. Atukorala, C. K. Chang, and K. Oyama. Situation-oriented evaluation and prioritization of requirements. In *Asia Pacific Requirements Engineering Conference*, pages 18–33. Springer, 2016.

[7] Y. Bengio and P. Frasconi. Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[8] K. K. Breitman, J. C. S. Leite, and A. Finkelstein. The world sa stage: a survey on requirements engineering using a real-life case study. *Journal of the Brazilian Computer Society*, 6(1):13–37, 1999.

[9] A. Casamayor, D. Godoy, and M. Campo. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4):436–445, 2010.

[10] C. K. Chang. Situation analytics: a foundation for a new software engineering paradigm. *Computer*, 49(1):24–33, 2016.

[11] C. K. Chang. Situation analyticsat the dawn of a new software engineering paradigm. *Science China Information Sciences*, 61(5):050101, 2018.

[12] C. K. Chang, H.-y. Jiang, H. Ming, and K. Oyama. Situ: A situation-theoretic approach to context-aware service evolution. *IEEE Transactions on Services Computing*, 2(3):261–275, 2009.

[13] R. Chapman, N. White, and J. Woodcock. What can agile methods bring to high-integrity software development? *Communications of the ACM*, 60(10):38–41, 2017.

[14] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[15] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. Automated classification of non-functional requirements. *Requirements Engineering*, 12(2):103–120, 2007.

[16] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287, 1988.

[17] A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2):3–50, 1993.

[18] T. Dietterich. Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*, pages 227–246, 2002.

[19] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski. Towards automated requirements prioritization and triage. *Requirements engineering*, 14(2):73–89, 2009.

[20] J. Eckhardt, A. Vogelsang, and D. M. Fernández. Are" non-functional" requirements really non-functional? an investigation of non-functional requirements in practice. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 832–842. IEEE, 2016.

[21] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. In *Situational awareness*, pages 9–42. Routledge, 2017.

[22] S. Feng, R. Manmatha, and A. McCallum. Exploring the use of conditional random field models and hmms for historical handwritten document recognition. In *Document Image Analysis for Libraries, 2006. DIAL'06. Second International Conference on*, pages 8–pp. IEEE, 2006.

[23] D. Firesmith. Using quality models to engineer quality requirements. *Journal of Object Technology*, 2(5):67–75, 2003.

[24] Z. P. Fry, B. Landau, and W. Weimer. A human study of patch maintainability. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pages 177–187. ACM, 2012.

[25] Z. P. Fry and W. Weimer. A human study of fault localization accuracy. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[26] Y. Gazi, M. S. Umar, and M. Sadiq. Classification of nfrs for information system. *International Journal of Computer Applications*, 115(22), 2015.

[27] S. Greenspan, J. Mylopoulos, and A. Borgida. On formal requirements modeling languages: Rml revisited. In *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on*, pages 135–147. IEEE, 1994.

[28] K. K. Gupta, B. Nath, and R. Kotagiri. Layered approach using conditional random fields for intrusion detection. *IEEE Transactions on dependable and secure Computing*, 7(1):35, 2010.

[29] N. Heumesser, A. Trendowicz, D. Kerkow, H. Gross, and L. Loomans. Essential and requisites for the management of evolution-requirements and incremental validation. *Information Technology for European Advancement, ITEA-EMPRESS consortium*, 2003.

[30] I. Im and B. H. Kim. Personalizing the settings for cf-based recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 245–248. ACM, 2010.

[31] D. Iren, G. Kul, and S. Bilgen. Utilization of synergetic human-machine clouds: a big data cleaning case. In *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering*, pages 15–18. ACM, 2014.

[32] C. Jensen and C. Potts. Experimental evaluation of a lightweight method for augmenting requirements analysis. In *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, pages 49–54. ACM, 2007.

[33] T. Keller. Contextual requirements elicitation: An overview. In *Seminar in Requirements Engineering, Department of Informatics, University of Zurich*, 2011.

[34] L. Krüger and B. Grabski. Using user context for accessing it resources. In *Proceedings of the first international workshop on Context-aware software technology and applications*, pages 33–36. ACM, 2009.

[35] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[36] D. Larlus, F. Perronnin, F. Meylan, P. Kompalli, and V. Mishra. Generating gold questions for difficult visual recognition tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop: Computer Vision and Human Computation*. Citeseer, 2014.

[37] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer. Genprog: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72, 2012.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] D. Lee and S. Helal. From activity recognition to situation recognition. In *International Conference on Smart Homes and Health Telematics*, pages 245–251. Springer, 2013.

[40] N. F. Lepora. Threshold learning for optimal decision making. In *Advances in Neural Information Processing Systems*, pages 3763–3771, 2016.

[41] S. Liaskos, S. McIlraith, and J. Mylopoulos. Representing and reasoning with preference requirements using goals. Technical report, Technical report, Dept. of Computer Science, University of Toronto, 2006.

[42] Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan. Protein fold recognition using segmentation conditional random fields (scrfs). *Journal of Computational Biology*, 13(2):394–406, 2006.

[43] A. Mahmoud. An information theoretic approach for extracting and tracing non-functional requirements. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 36–45. IEEE, 2015.

[44] A. McCallum, D. Freitag, and F. C. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

[45] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems (TOIS)*, 8(4):325–362, 1990.

[46] P. Neumann. 681 topic report. 2004.

[47] B. Nunnally and D. Farkas. Ux research: Practical techniques for designing better products. 2016.

[48] N. Okazaki. Crfsuite - crf benchmark test. http://www.chokkan.org/software/crfsuite/benchmark.html.

[49] N. Ponomareva, P. Rosso, F. Pla, and A. Molina. Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*, pages 479–483, 2007.

[50] A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. *KR*, 91:473–484, 1991.

[51] S. Robertson and J. Robertson. *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.

[52] G.-C. Roman. A taxonomy of current issues in requirements engineering. *Computer*, (4):14–23, 1985.

[53] J. Rowley. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of information science*, 33(2):163–180, 2007.

[54] K. Sato and Y. Sakakibara. Rna secondary structural alignment with conditional random fields. *Bioinformatics*, 21(suppl_2):ii237–ii242, 2005.

[55] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE network*, 8(5):22–32, 1994.

[56] V. S. Sharma, R. R. Ramnani, and S. Sengupta. A framework for identifying and analyzing non-functional requirements from text. In *Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture*, pages 1–8. ACM, 2014.

[57] J. Slankas and L. Williams. Automated extraction of non-functional requirements in available documentation. In *Natural Language Analysis in Software Engineering (NaturaLiSE), 2013 1st International Workshop on*, pages 9–16. IEEE, 2013.

[58] I. Sommerville and P. Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.

[59] N. Subramanian and L. Chung. Tool support for engineering adaptability into software architecture. In *Proceedings of the International Workshop on Principles of Software Evolution*, pages 86–96. ACM, 2002.

[60] G. M. Sullivan and R. Feinn. Using effect sizeor why the p value is not enough. *Journal of graduate medical education*, 4(3):279–282, 2012.

[61] C. Tibermacine, R. Fleurquin, and S. Sadou. Nfrs-aware architectural evolution of component-based software. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 388–391. ACM, 2005.

[62] A. Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[63] A. van Lamsweerde. Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 4–7. IEEE, 2004.

[64] A. Van Lamsweerde. *Requirements engineering: From system goals to UML models to software*, volume 10. Chichester, UK: John Wiley & Sons, 2009.

[65] L. Xiao. Intrusion detection using probabilistic graphical models. 2016.

[66] H. Xie, J. Yang, C. K. Chang, and L. Liu. A statistical analysis approach to predict user's changing requirements for software service evolution. *Journal of Systems and Software*, 132:147–164, 2017.

[67] J. Yang, C. K. Chang, and H. Ming. A situation-centric approach to identifying new user intentions using the mtl method. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, volume 1, pages 347–356. IEEE, 2017.

[68] S. S. Yau, H. Gong, D. Huang, W. Gao, and L. Zhu. Specification, decomposition and agent synthesis for situation-aware service-based systems. *Journal of Systems and Software*, 81(10):1663–1680, 2008.

[69] E. Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.

[70] E. S. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE, 1997.

[71] E. S. Yu and J. Mylopoulos. From er to" ar"-modelling strategic actor relationships for business process reengineering. In *Proceedings of 13th Int. Conf. on the Entity-Relationship Approach (ER'94), number 881 in Lecture Notes in Computer Science*. Citeseer, 1994.

[72] E. S. Yu and J. Mylopoulos. Understanding" why" in software process modelling, analysis, and design. In *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on*, pages 159–168. IEEE, 1994.

[73] N. Yusop, D. Zowghi, and D. Lowe. The impacts of non-functional requirements in web system projects. *International Journal of Value Chain Management*, 2(1):18–32, 2007.

[74] W. Zhang, Y. Yang, Q. Wang, and F. Shu. An empirical study on classification of non-functional requirements. In *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, pages 190–195, 2011.

[75] L. Zhu and I. Gorton. Uml profiles for design decisions and non-functional requirements. In *Proceedings of the Second Workshop on Sharing and Reusing Architectural Knowledge Architecture, Rationale, and Design intent*, page 8. IEEE Computer Society, 2007.

[76] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang. Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pages 446–449. IEEE, 2015.

APPENDIX. COOPERATIVE RESEARCH ENVIRONMENT

The Cooperative Research Environment (CoRE) is generally used for researchers to share their thoughts and views on academic papers. A registered member in the CoRE system can view all the papers and comments, download papers, view profile under the menu "My account". Besides, he or she can also upload/edit a paper and submit/edit a comment for a paper. The detail operations a user can perform on the CoRE system are listed below.

Log in/Log out: To login the CoRE system, a user can access the login page by clicking the link "log in" under the menu "My account". Type in the login ID and password to login the system. After logging in, there will be a "log out" link under the menu "My account" from which a user can log out.

View your profile: A user can access the profile page by clicking the link "My profile" under the menu "My account". On the profile page, there is a form showing user profile, including the login ID, roles, number of papers the user uploaded and number of comments the user submitted.

View all papers list: A user can access the page "list of all papers" by clicking the link "View papers and comments" under the menu "My account". On the page "list of all papers", there is a form showing all the uploaded papers. Each row of the form shows one paper's information and operational links associated with the paper.

View your uploaded paper list: A user can access the page "list of my uploaded papers" by clicking the link "My uploaded papers" under the menu "Member". Similar to the page "list of all papers", the user can see a form showing his or her uploaded papers on the page "list of my uploaded papers".

View your commented papers: A user can access the page "list of my commented papers" by clicking the link "My commented papers" under the menu "Member". Similar to the page "list of all papers", the user can see a form showing his or her commented papers on the page "list of my commented papers".

View a paper's information and comments: A user can view the information and comments of a paper by clicking the title of the paper on the page "list of all papers", "list of my uploaded papers" or "list of my commented papers".

Download a paper: A user can download a paper by clicking the link "Download" in the same row of this paper on the page "list of all papers", "list of my uploaded papers" and "list of my commented papers".

Create filters for paper list: A user can create filters for the list of papers showing on the page "list of all papers". By clicking the link "show the selection form", the user will see the selection form which shows all available filters: title, authors, publication type, and publish date, etc. Customize the filters, and then click the button "Filter" to update the list of papers. The list of papers showing in the form will meet all the filters the user input or select in the selection form. Please note: for the filter "categories", if no category is checked, papers will be selected from "any category"; and if several categories are checked, papers will be selected from "any one in these categories".

Make a comment: If a user has not submitted any comment for a paper, the user can see a "Comment" link in the row of a paper on the page "list of all papers", "list of my uploaded papers" and "list of my commented papers". Click the link "Comment", and the user will see a comment submission form. In the form, the fields "Problem solved" and "Discussion" require at least 40 words, and the fields "Methodology", "Technical merits" and "Limitation" require at least 100 words.

Edit a comment: If a user has submitted a comment for a paper, the user can see an "Edit comment" link in a row of a paper on the page "list of all papers", "list of my uploaded papers" and "list of my commented papers". Click the link "Edit comment", and the user will see a comment submission form with his or her previous comment, and the user can edit his or her comment and submit the revision.

Remove a comment: If a user has submitted a comment for a paper, the user can see a "Remove comment" link in a row of a paper on the page "list of all papers", "list of my uploaded papers" and "list of my commented papers". Click the link "Remove comment", and the user will see a pop-up dialogue asking him or her to confirm the removal action.

Upload a paper: A user can access the page "upload a paper" by clicking the link "Upload a paper" under the menu "Member". The user can see a paper upload form on the page "upload a paper", on which he or she can input or select paper information, and choose a file to upload. After finishing inputting all paper information, the user need to click button "Comment >>" to submit a comment for this paper. If the paper information and comment information satisfies all the requirements, the paper and comments will be accepted by the system.

Edit a paper: If a user is the uploader of a paper, the user will see an "Edit paper" link in the paper's row on the page "list of all papers", "list of my uploaded papers" and "list of my commented papers". Click the link "Edit paper", and the user can edit the information of the paper.