A Multi-layered Desires Based Framework to Detect Users' Evolving Non-functional Requirements

Peng Sun
Department of Computer Science
Iowa State University
Ames, USA
psun@iastate.edu

Jingwei Yang
Department of Computer Science
James Madison University
Harrisonburg, USA
yang6jx@jmu.edu

Hua Ming
Department of Computer Science & Engineering
Oakland University
Rochester, USA
ming@oakland.edu

Carl K. Chang
Department of Computer Science
Iowa State University
Ames, USA
chang@iastate.edu

***Abstract*—Non-functional requirements (NFRs) play a crucial role in all the downstream activities of a software life-cycle process. Capturing newly emerged NFRs is key to software evolution. Recent research shows functional requirements in the form of task-level alternative features can be elicited from user behavioral and system contextual data through user goal inference. Considering the close connection between the concept of goal and desire, we posit that there is an opportunity to extract new NFRs based on users' mental states, particularly their desires. We propose to use a statistical model to infer desires with multiple-levels of abstraction based on contextual data under Situ framework. Our multi-layered desire inference method takes inference confidence into consideration, and tries to make sense of inference results with both high- and low- inference confidence. By utilizing the different abstraction levels of desires, we provide an illustrative example with three cases to elicit users' new NFRs including new high-level and low-level desires and new contributing relationships between them. Several implications of this work are also discussed. We plan to conduct experiments on human subjects to validate the proposed method as IRB has just approved our proposal.**

***Index Terms*—Conditional Random Fields, Desires Inference, NonFunctional Requirement, Software Engineering**

I. Introduction

As the early-phase of software requirements, Non-Functional Requirements (NFRs) elicitation is a crucial part of the software system requirements process. Approximately 50-60% of the errors in software development are caused by incorrect requirements, which increases the cost of fixing errors up to 100 times more in the later phases of the software development process [23]. NFRs emphasize on how to analyze and address stakeholder's interests from a series of system design alternatives [35]. A good understanding and analysis of NFRs can help make better design decisions [39], requirements changes [16], testing plans, and architectural evolution [27], [28] etc. New NFRs should be promptly discovered and properly addressed in the evolution process of a software product [35].

Manually eliciting NFRs is labor-intensive and time-consuming. One line of work approaches the problem of NFRs extraction with Natural Language Processing (NLP) techniques based on textual requirements specifications [22], [25], user manuals and data agreements [26]. NLP based approaches often assume the requirement documentations are well articulated; however, NFRs are generally poorly presented in the requirement documents [1], [15], [37], and many real-life software products lack those well documented requirement specifications due to the non-robust documentation techniques, which make mining NFRs from text contents difficult to achieve in practice. Furthermore, a recent study by Eckhardt et al. [11] reported that about 75% of so called system-specific NFRs are not non-functional based on a classification of 530 NFRs extracted from eleven industrial requirements specifications. This implicates the limitation of using requirement specifications to elicit NFRs.

Traditional requirements elicitation methods primarily deal with human stakeholders, inevitably making some of the acquired user requirements not completely objective, which may eventually lead to future system imperfectness or even faults [11]. Alternatively, user behavioral and system contextual data are considered largely objective, in the sense that they can reflect how users interact with the system, and how the system responds in real-world execution scenarios. Thus, eliciting user requirements from observation centric raw data could be a good compensation to traditional methods, which will help capture more complete user requirements to evolve the system. Recent research shows functional requirements in the form of task-level alternative features can be elicited from user behavioral and system contextual data through user goal inference [17], [32], [33]. We believe that human mental states, such as desires and intentions, are closely related and can be mapped to system goals, and there is an opportunity to extract high-level desires in the form of NFRs from contextual data.

We propose to analyze NFRs within a multi-layered desires framework, eliciting new relations among desires in different layers, by considering desires inference confidence. Our method is built under Situ framework [4], [5], which has been used to analyze new system requirements based on goals tree [5]. We believe that desires should have a hierarchical division such that users' concrete mental states can be represented as a desire tree. Situ framework supports human mental states (desires) which helps map desires to NFRs since the natural characteristics of NFRs are subjective, abstract [3], [24], relative, interacting [7], and diverse [13]. In addition, a situation sequence could represent how a user wants to accomplish their interests with time sequential observations; thus, we consider that the Situ framework with multi-layered desires has the potential to discover users' evolving NFRs. By reflecting upon the current methods for user goal inference and new intentions identification in the Situ framework [32], we feel the increasing needs to fill the void of NFRs in Situ.

This paper extended Xie at al.'s work [32] of using Conditional Random Fields (CRFs) to elicit requirements. The advantages of our method over Xie's is that we progressed further from goals inference to genuine human desires inference, and we treat desires as multi-layered, and our method provides a quantified-guideline for domain experts to reason and analyze users' new needs. We discuss the relationship between the concepts of goal and desire and propose to use a statistical model, Conditional Random Fields (CRFs), to infer human desires and to elicit users' new NFRs. The proposed method is expected to be semi-automatic, which will help improve engineers productivity and workflow efficiency during NFRs elicitation process. As described in Section V, step 1, 2, and 6 are semi-automatic, and step 3, 4, and 5 are automatic. The Situ based NFRs analysis would be one step further toward the run-time system evolving envisioned in the work of Situation Analytics [4].

The contributions of this work are:

- Propose a method to infer users' genuine multi-layered desires with CRFs model based on the observation of their behavior and environmental context.
- Propose three detailed NFRs elicitation methods based on desire inference confidence to discover new desires and new contributing relations among different abstraction level desires
- Present results in a preliminary illustrative example, and illustrate how to use our method to elicit new NFRs for a real-life application.
- Discuss the promising impact of our method on system design, system architectures, individualized requirement analysis, etc.

The rest of this paper is organized as follows. Section II provides some background information about NFRs, Situ framework and sequential learning model. A running example is introduced in Section III to assist interpreting the method. The comparison between desires and goals as well as the relation between desires and NFRs are discussed in Section IV. Sections V describes the method to infer desires and elicit NFRs. A illustrative example is presented in Section VI followed by the implication of this work (Section VII), the future work (Section IX), and conclusion (Section X).

II. Background & Related Work

A. NFRs

One line of recent work has studied the NFRs from various perspectives, such as Model-Driven Development [2], distinguishing NFRs from functional requirements (FRs) [11], [18], or NFRs automatic reasoning tools [31]. In addition to the NFRs elicitation approaches, goal oriented models such as i* model [36], and Kaos [29] have been provided for requirement engineers to evaluate constraints and tradeoffs of NFRs. Based on these models, stakeholders could have structured ways to brainstorm and document NFRs from a system-centered perspective. However, those models cannot capture all the evolving needs of stakeholders. To compensate for that, we focus on NFRs elicitation specifically from the human-centered perspective.

NLP and text mining techniques have been applied as a very popular approach for NFRs elicitation [8], [22], [25], [38]. Cleland-Huang et al. proposed an automated approach to extract NFRs from various software development documentations. They assumed that keywords could be used as indicator items to distinguish different types of NFRs. Those keywords need to be manually mined from documentation and used as input to train a classifier, which is later used for the NFRs classification. Based on a evaluation of 30 students software projects, a very high false-positive rate was found [8] .

One work study developer-centered NRFs is proposed by Zou et al. [40]. They use topic model to mine text contents from Stack Overflow posts and discussions in order to elicit developers' NFRs. However, developers cannot represent users with respect to NFRs, and it often requires an undesirable long period of time to collect enough posts for analysis when using an online discussion forum. Moreover, their technique cannot understand the real meaning of the text from discussion. In contrast, by monitoring system users contiguous mental states, behaviors, and environmental context information, we have a better chance to capture the much more updated, real-time NFRs from users' perspectives.

B. Situation

Situation awareness outperforms context-awareness by considering more from human-centric dimensions. The research community of situation awareness has defined a situation in various ways [5], [12], [20], [34]. In this paper, we apply the definition given by Chang et al. [5], which considers a situation as a three-tuple including human mental states, behaviors, and environments, because it provides a good abstraction that contains the necessary elements to compute a situation from the computer science viewpoint. [32].

The software engineering community has already probed requirements with situation analysis. Xie et al. applied situation framework on requirements elicitation [32], and they employed CRFs to build the relations between user observations and the

corresponding goals. However, we argue that not all system goals are at the same abstraction level. Neither are user desires, and we believe that CRFs could be used to infer human desires at different abstraction level. Yang et al. [33] proposed to use a Multi-strategy Task-adpative Learning method (MTL) on top of the inference result by CRFs to automate the process of identifying new requirements in the form of intention. However, it highly depends on the accuracy of CRFs learning, and cannot handle noise very well. On the contrary, in our method we consider the scenarios when CRFs cannot properly identify a predefined desire for a situation due to lack of domain knowledge and training data, and the confidence level of inference result is low.

C. Sequential supervised learning models

To deal with sequential data, some supervised learning algorithms have been studied such as sliding window methods, graph transformer networks, hidden Markov models, conditional random fields, etc. [10]. Unlike some classical classification framework in which the training data is draw identically and independently from joint distribution, those nearby data in the sequence have significant correlations [10]. Thus, it is crucial to find an appropriate model that can capture relations among each single observation and predicted label in the sequence. Some methods, such as sliding window methods and hidden Markov model, cannot represent the relation of among nearby observations and labels. CRFs has the advantage of providing flexible features templates to generate feature functions that can be used to represent relations among correlated sequential data. The detailed reasons for using CRFs are discussed in Section V-A.

III. A Running Example

To illustrate, we introduce a real world like software system, Cooperative Research Environment (CoRE) [32] to help us explain concepts and methodology. CoRE is generally used for researchers to share their ideas and views on academic papers. Some operations that users could perform on this system include view all the papers and comments, download and upload papers, edit user profile. Also, users can upload/edit a paper and submit/edit a comment for a paper. We will use those operations mentioned above to clarify desires and NFRs in this software system domain. Figure 1 shows the interface where users can upload a paper. The assumption of our method is that there is an existing version of system/prototype available for data collection. Also there is some domain knowledge (maybe incomplete) and training data available. The CoRE system satisfies our assumption.

IV. Desire vs. Goal

In this section, we differentiate the concepts of desires and goals. We also describe how to elicit and model NFRs from different abstraction levels of desires.

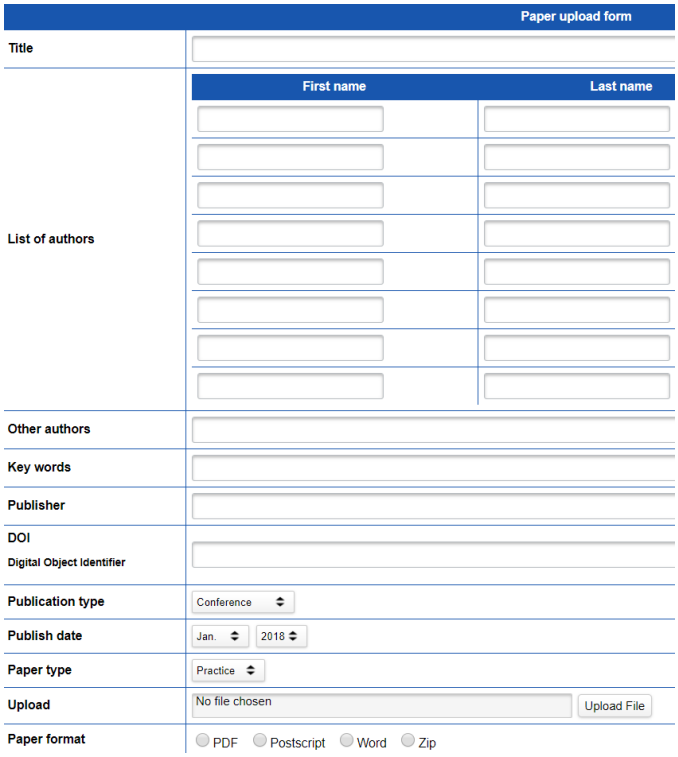

Fig. 1. Upload a paper

A. Goals

In goal-oriented requirement engineering (GORE), the concept of a goal is interweaved with scope and stakeholders, such that its achievement requires the cooperation of multiple agents, which cannot be isolated from one another [9], [23]. Lamsweerde gave the definition of a goal in this way [30]:

"A goal is a prescriptive statement of intent that the system should satisfy through the cooperation of its agents".

Humans, as one kind of system agent, play an important role in goal satisfaction by interacting with a software system. The literature of GORE offers a bridging method via binding a goal from human users with the system under development, which does not support eliciting requirements truly from human perspective irrespective of the system perspective. In this work, we explore human requirements by dealing with genuine human desires with regard to their mental state. System perspectives become a lesser concern or even absent.

B. Desires

We differentiate desires and goals by annotating that desires and goals are at different levels with respect to a human's mental state. Desires are usually along with human cognition, emotions and moods, which are at a much higher level compared to goals that bind system perspectives. As explained before, our desire-oriented approach primarily focuses on human desires by exploring human mental states. For example, in CoRE a goal could be described as "users should be able to

submit comments in the system," whereas a desire of a certain user in the system could be stated as "I want to discuss with somebody". We study NFRs from the perspective of desires because desires could have a better reflection of human mental states, including what users really want and need, compared with goals.

In the situation awareness computing literature, researchers [5] have proposed to represent desires with a tree structure, which is similar to multi-layered goals structure. Similar to the goals, we divide desires into multiple hierarchies with different abstraction levels. A low-level desire denotes a user's eagerness for achieving a desirable outcome. A high-level desire could consist of multiple low-level desires. Figure 2 describes a high-level desire, "find a collaborator." This could be decomposed into two low-level desires such as "discuss" and "share a work." The relations among multi-layered desires can indicate how users want to address their interests, which is implicit the NFRs. For instance, in order to realize the desire "find a collaborator," a user might first need to share a work and then discuss with somebody in the system.

Desires are difficult to capture. However, the rapid development of fields such as brain informatics, affective computing, and psychophysiology bring the opportunity for software engineers to tackle human desire during the development and maintenance of a software system [4]. Moreover, software engineers could take advantage of the techniques such as gesture-recognition, eye-tracking, and electroencephalogram devices within sensor-laden environments to track desires once they are linked to actual actions [4].

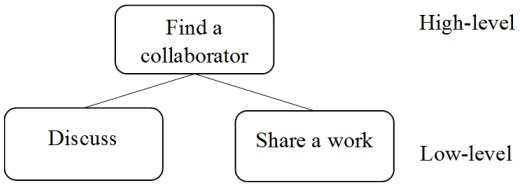

Fig. 2. Desire decomposition

C. NFRs in Desires

We note that there may be multiple ways for a user to achieve a high-level desire through low-level desires, which are indicative of quality of services, and imply NFRs from human perspectives. For example, Figure 3 shows that in order to realize a high-level desire "build reputation," a user may first need to achieve a low-level desire "share a work." There are two different ways through which a user could achieve the desire "build reputation" via "share a work"; the user can either share a work as detailed as possible or share a work as conveniently as possible. In the former case, a user could provide as much information as possible when uploading papers to the system, which has a positive contribution towards realizing the high-level desire; for the latter case, a user may prefer convenience and omit some information when uploading papers, which has a negative contribution to achieving the high-level desire. Though both positive and negative contributions exist, the links from low-level to high-level desire in our multi-layered desires structure only represent positive contributions. This is due to the nature of our desire inference method, which will be described in the following section.

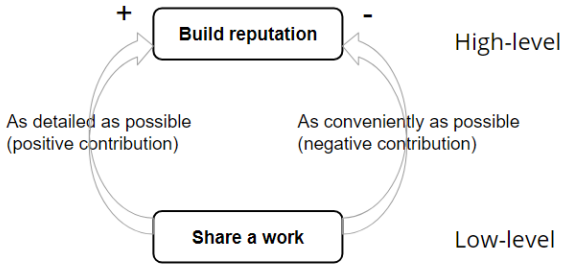

Fig. 3. NFRs among desires

V. Methodology

In this section, we introduce how to elicit new NFRs from observation sequences with a statistical model CRFs.

A. Why CRFs

Human desires change over time along with human actions and system environmental contexts. Chang [4] proposed a situation analysis framework that considers a situation as a time-stamped organic whole of human mental states, behavioral contexts, and environmental contexts. Our problem fits well under situation framework because desire is hidden in human mental states and can be inferred by external behaviors and environments. Researchers in requirements engineering have successfully applied CRFs to infer goals [32]. We believe that CRFs could be an appropriate model for desire inference because 1) both desires and goals can be represented as tree structures, and low-level desires are usually very concrete as for users, so they can be essentially mapped to low-level system goals; 2) goals can serve as requirements to assist users to realize their desires [5], so both low-level desires and goals are contributing to high-level desires. From this point of view, CRFs is promising for low-level desire inference.

We believe that CRFs are not only suitable for underpinning low-level desires, but also work for more abstract high-level desires. There are several reasons for this: 1) high-level desires are similar to low-level desires from the aspect that both of them represent human mental states and have external reflection in the form of behavior sequences under specific contexts; 2) the process of a user realizing a high-level desire from achieving several low-level desires is just like how a user realizing a single desire from several actions. Thus, high-level desires could be inferred in the same way as low-level desires, considering only observations, or inferred with the combination of low-level desires and observations. In this paper, we use only observations to infer every level of desire because currently we do not possess a usable data set for running inference from conducting an experiment under development. Our previous IRB-approved experiment was not set in the multi-layered desire-inference framework.

CRFs has been proven to outperform HMM (Hidden Markov Models) and MEMM (Maximum Entropy Markov Model) in accuracy for tagging sentences in part-of-speech (POS) problem [10]. Besides labeling POS, CRFs has also been applied on biological sequences data [21], pattern recognition and machine learning [19] , and intrusion detection [14]. Compared to HMM, CRFs is more appropriate to interpret the relations among desires, actions, and system context values since it can overcome the limitations of HMM by using the flexible feature functions instead of a probability matrix. Some existing popular open-source tools of CRFs are CRF++ written in C++, CRFsuite written in C, Pycrfsuite written in python, etc.

We emphasize that the effort to apply CRFs will be reasonable for the reasons that 1) training data, which is inevitable for any supervised learning model, can be recorded by a embedded monitoring mechanism, such as computer vision, easily ignoring specific software applications; 2) the attempts to define feature templates can be no more than trying unigram, bigram, or both of them, and in most cases, the templates are defined by considering several consecutive observations, so feature templates can be reused across systems conveniently. Thus, we argue that it is not difficult to use CRFs in real-life software application.

B. NFRs elicitation with CRFs

Our method of NFRs elicitation takes two steps: 1) using the CRFs model to infer desires; 2) analyzing relationships among desires at different abstraction levels to elicit NFRs. Our method infers desires from observations, which are time stamped statuses of a system domain, including user actions and system environmental context values. Figure 4 presents the workflow of our method. First, domain experts train a CRFs model, then the CRFs model takes observation data as input, and outputs inferred desires with confidence values. Experts, using the inference confidence can discover new contributing relationships between different levels of desires, or can decide which portion of observation sequences need to be further analyzed for new desires. New contributing relationships and new desires can be used to elicit new NFRs, which will be added to the initial knowledge base and interact with existing domain knowledge. The detailed method is described as follows:

1) Building Initial Knowledge Base: As a supervised learning method, a CRFs model should be built on the existing domain knowledge, so the input of training data generally comes from domain experts or proficient system users. The format of training data is a set of observation sequences combined with 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), should be collected from observing 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 self-reported by proficient users or labeled by domain experts accurately. $(d_{ai}, d_{bi}, d_{ci}, ...)$ represents desires of different abstraction levels, with d_{ai} as the lowest one.
2) Training the CRFs Model: First, we need to define feature functions or feature templates in the same way as [32]. Note that for each abstraction level of desires, a corresponding set of feature functions/templates needs to be defined independently. Then by feeding the training data to feature functions/templates, a CRFs model can be trained.
3) Data Collection and Pre-processing: Observations of users' operations on the target system shall be collected. Specifically, users' actions and system context values will be recorded. The observation sequences,$< o_{11}, o_{12}, ..., o_{1j} >, < o_{21}, o_{22}, ..., o_{2k} > ...$, will be pre-processed to the required format.
4) Desires Inference: Once the CRF model and collected observations are ready, we apply the model on the observation sequences to assign inferred desires, $(\hat{d_{ai}}, \hat{d_{bi}}, \hat{d_{ci}}, ...)$, for each observation. Similar to 2), the inference processes of desires at different abstraction levels are independent from each other.
5) Situation Partition: For every two consecutive levels of desire, such as $(\hat{d_{ai}}, \hat{d_{bi}})^*$, from the lowest to the highest level in a desires sequence $(\hat{d_{ai}}, \hat{d_{bi}}, \hat{d_{ci}}, ...)^*$, set thresholds of inference confidence and divide inferred desires into relatively high confident and low confident ones, and map situations to four cases described in Table I based on their desires' inference confidence.
6) Exploring Relations Between Two Different Abstraction Level Desires for New NFRs: Regarding desires in neighboring levels, such as $(\hat{d_{ai}}, \hat{d_{bi}})^*$, for each case i in Table I, domain experts focus on its inclusive situations whose appearance frequency is larger than a predefined, domain-specific threshold f_i, to locate data of interests. By analyzing their surrounding situation sequences, experts shall look for new relations between two adjacent levels of desires. As we discussed in Section IV-C, those new relationships implicate the emerging NFRs (see details in Section VI-D).

TABLE I
MEASURES USED FOR EVALUATION.

		High-level Desire	
		High Conf.	Low Conf.
Low-level Desire	High Conf.	hh	lh
	Low Conf.	hl	ll

(Conf. represents Confidence)

VI. AN ILLUSTRATIVE EXAMPLE

In this section, we illustrate how to apply our methodology using the CoRE system. CoRE is modified from an open source, manuscript management system, "MyReview"[1]. The modification kept its basic functionalities, and added necessary

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

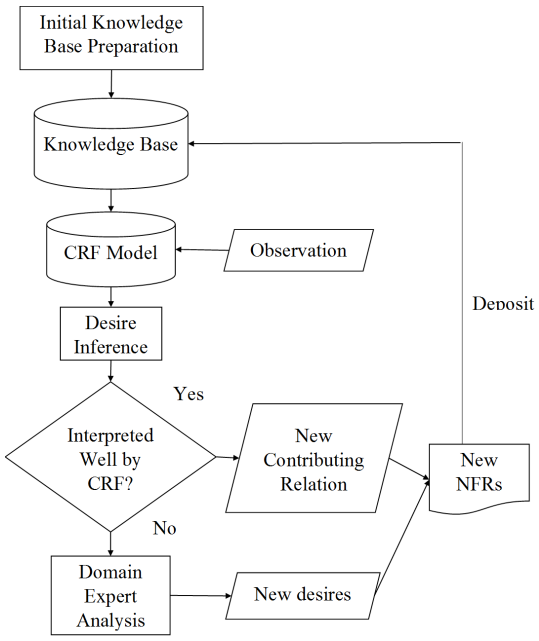

Fig. 4. Work flow of NFRs elicitation

instrumental code to record user behaviors and system context variables. More details about CoRE and their previous experiment can be found in [32]. In this illustrative example, our discussion mainly centers on the system function for uploading a paper (Figure 1). When a user wants to achieve the desire "share a work," the user needs to upload a paper by providing its information such as the paper title, authors name, key words, etc., and upload its electronic version as a PDF file.

A. Initial Knowledge Base Construction

To acquire the training data, it requires domain experts to operate on CoRE and label their desires as high-level or low-level for each action they take. To keep our discussion simple, we only consider two levels of desires: high- and low- level desires. Table II shows the input training data, with the first column as time points, followed by a situation sequence to achieve the high-level desire "build reputation" in the second column, and another situation sequence to achieve a different high-level desire "find a collaborator" in the third column. We note that each situation S_i consists of an observation and a desire pair such as $< O_i, (D_l, D_h)_i >$, and for each observation O_i it consists of expert action and related system context information, which will be separated from the action by "&". For example, in the second column, the situation at time point 2 describes that an expert clicked the "Submit" button, and the system feedback was that the operation was successful, operation duration was 20s, and the expert's low- and high-level desires are "share a work" and "build reputation" respectively.

B. Feature Functions

We can use both bigram and unigram feature templates [32] in CRF++ to train models for high- and low- level desires inference separately.

C. Desire Inference

Once the training data and feature templates are ready, we can feed them to CRF++ for training. The training process of a CRFs model took several iterations until the improvement of the log likelihood over the last iteration is not greater than a threshold, which is set as a default value by CRFs tools.

After the model is generated, observations on ordinary users' real world behaviors and related system context values can be collected. The collected data should be pre-processed to have the same format with the training data, and are fed to CRF++ with inference model trained to infer desires. The output is a file containing observations followed by predicted desires and an inference confidence value. Table III presents several examples of observations with desires inferred with different abstraction levels, as well as different inference confidence.

D. NFRs Elicitation

In Section IV-C, we pose NFRs among desires. A NFR could be defined through an alternative that specifies how a user's interest is addressed, specifically how to realize a higher-level desire through multiple lower-level desires in a specific way. Users have several alternatives to achieve a high-level desire from low-level desires. Those alternatives specifying how users want to achieve their desires could be used for the NFRs analysis.

Once the high- and low-level desires are inferred, domain experts can study the relations between different abstraction levels of desires to analyze NFRs. In this section, we describe several approaches to representing NFRs based on the inference confidence of desires.

Divergent behaviors could be a root cause of low confidence desires inference because those behaviors generally happen when users act in unexpected ways, which are not included in the initial knowledge base. It is possible that users have some mistaken operations on the system that make their desire hard to interpret; however, if this sort of situation frequently occurs, it is more likely that users have emerging new desires that the existing knowledge base does not have. Therefore, those situations with uninterpreted desires are an important resource to domain experts for further analysis.

Based on desires inference results, we can divide them into groups by inference confidence. As presented in Table I, both high- and low-level desires are partitioned into two sections, high confidence and low confidence desires. In practice, domain experts should decide thresholds, P_{lt} and P_{ht}, to set the partitions of confidence for different abstraction levels of desires since those thresholds are usually domain specific. In our case, the confidence of desires formulates four combinations described in Table I, and we will analyze each case for NFRs elicitation respectively.

1) High confidence high- and low-level desires (hh): In this case, both high- and low-level desires are inferred with a high confidence, indicating that both level desires very likely exist in the desires list of the training set and can be interpreted accurately by the CRFs model. The low-level desires inferred shall be contributing to the high-level desire since we assume

TABLE II

TRAINING DATA EXAMPLES: TWO SITUATION SEQUENCES (d_{h1}: BUILD REPUTATION, d_{h2}: FIND A COLLABORATOR, d_{l1}: SHARE A WORK, d_{l2}: MODIFY PERSONAL INFORMATION, AND d_{l3}: DISCUSS)

Time point	$< O_1^*, (D_l, D_h)_1^* >$	$< O_2^*, (D_l, D_h)_2^* >$	...
1	(clickMenuUploadPaper & (duration: 18s), (d_{l1}, d_{h1}))	(clickMenuAllPapers & (duration: 27s), (d_{l3}, d_{h2}))	
2	(clickSubmit & (success; duration: 20s), (d_{l1}, d_{h1}))	(clickComment & (duration: 10s), (d_{l3}, d_{h2}))	
3	(clickMyProfile & (duration: 6s), (d_{l2}, d_{h1}))	(clickSubmit & (CommentGood; duration: 8s), (d_{l3}, d_{h2}))	
4	(clickUpdate & (success; duration: 3s), (d_{l2}, d_{h1}))	(clickMenuUploadPaper & (duration: 30s), (d_{l1}, d_{h2}))	
5	(clickMenuUploadPaper & (duration: 17s), (d_{l1}, d_{h1}))	(clickSubmit & (success; duration: 2s), (d_{l1}, d_{h2}))	
6	(clickSubmit & (success; duration: 1s), (d_{l1}, d_{h1}))		
...	...	...	

TABLE III

INFERENCE RESULT FOR INPUT OBSERVATIONS (P_{lt} AND P_{ht} ARE CONFIDENCE THRESHOLD FOR LOW- AND HIGH-LEVEL DESIRES)

Case	Action	Context Value	Inferred Desires (Confidence)	
			Low-level	High-level
hh	clickComment	Page_comment	discuss ($P_l > P_{lt}$)	build reputation ($P_h > P_{ht}$)
lh	clickMenuUploadPaper	data context: abbreviated operation duration: short	share a work ($P_l > P_{lt}$)	anyone ($P_h < P_{ht}$)
hl	clickEditPaper	Page_EditPaper	anyone ($P_l < P_{lt}$)	find a collaborator ($P_h > P_{ht}$)
ll	clickMyProfile	data context: abbreviated operation duration: short	anyone ($P_l < P_{lt}$)	anyone ($P_h < P_{ht}$)

users are rational. If this contributing relation in the low- and high-level desires pair does not exist in the training data, it means a new contributing relation emerges, and we can add it to the existing knowledge base. For example, case hh in Table III, two level desires are inferred with probability larger than predefined thresholds, and if this phenomenon occurs frequently, it indicates that the low-level desire "discuss" is contributing to the high-level desire "build reputation." After consulting with domain experts, it is confirmed that discussion could be a way to build reputations. More importantly, when domain experts may not consider all the relations between different abstraction levels of desires, case hh in Table III shows the opportunity to automatically discover and add new contributing relations into the initial knowledge base.

2) Low confidence high-level desires and high confidence low-level desires (lh): Domain experts can start from a situation wherever a high-level desire has a low inference confidence, whose probability is less than a domain specified threshold P_{ht}, and the corresponding low-level desire has a high inference confidence, whose probability is larger than a predefined threshold P_{lt}. We interpret this case as that the inferred low-level desire exists in the training data, while the inferred high-level desire does not. In other words, we believe that the low-level desire is contributing to some "unknown" high-level desire that has to be further investigated by domain experts. To reveal such an "unknown" desire, domain experts can examine those related observation (situation) sequences for useful hints. For example, in case lh, from the user's behavior data, it appears that the user was submitting a paper. However, the related contextual data shows that user's operation duration was short and their submitted paper information was abbreviated. Such pattern implies that the user has tried to provide the required information succinctly for convenience on the page of "Upload a paper" shown in Figure 1.

Thus, a new alternative of low-level desire "share a work as conveniently as possible" has been identified. In fact, the initial desire "share a work" in the training data is meant to be "share a work as detailed as possible," with positive contribution to high-level desire "build reputation."

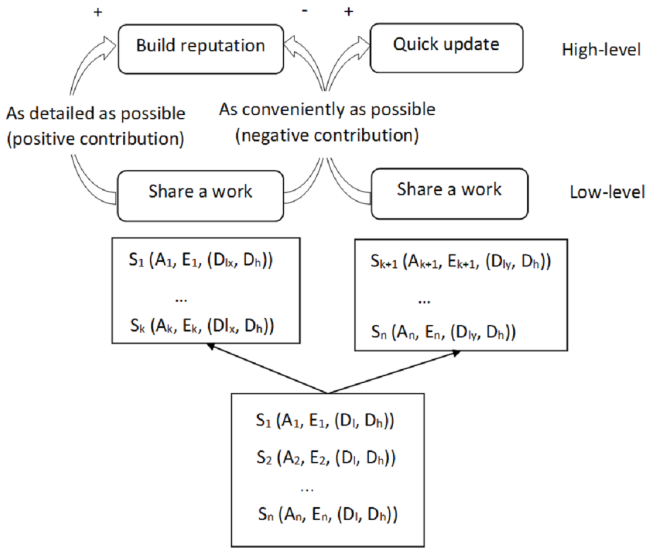

Fig. 5. Discover new high-level desire (D_{lx} represents the desire to share a work as detailed as possible; D_{ly} represents the desire to share a work as conveniently as possible)

As for the "unknown" high-level desire, one possible explanation is that users wanted to give a "quick update." If so, the user may intend to skip the tedious process of providing all information, but provide only some key information when uploading a paper. As presented in Figure 5, the low-level desire "share a work" has positive contribution to the new high-level desire "quick update" if a user choose to share a work as conveniently as possible. This newly identified high-level desire

"quick update" can be added to the existing multi-layered desire relationship network, and will interact with related existing desires. Such new "desire" interaction may further influence the overall system design [33].

3) High confidence high-level desires and low confidence low-level desires (hl): To deal with the reversed situation of the last case (lh), we can pay attention to situations that are predicted with a high confident high-level desire and a low confident low-level desire. This happens when there are some new low-level desires that are contributing to the existing high-level desire, which implies new alternative for that high-level desire. For example, in case hl, the action is to click button to edit a paper, and the CRFs model cannot interpret the low-level desire largely due to the lack of certain knowledge in training data that corresponds to the user's genuine low-level desire. Then, it may imply that there may be some other low-level desire, not in the low-level desire list of training data, that could be contributing to the high-level desire "find a collaborator". Domain experts could reason the possible new low-level desires based on this or surrounding situations. Further investigation on the actions and system contexts of case hl may indicate that the user want to edit a paper that previously uploaded in the system, so a new alternative to find a collaborator may be "edit shared work". Through this perspective, new alternatives to achieve existing high-level desires may be discovered.

4) Low confidence high- and low-level desires (ll): If the output of CRFs for a observation is that both high- and low-level desires are inferred with low confidence, less than the confidence thresholds (P_{lt} and P_{ht}), it is likely that users' real desires (both high- and low-levels) are not incorporated in the existing knowledge base. This is the most difficult case among all four cases because there are no reliable neighboring desires to rely on for further analysis, like the other three cases. For example, case ll in Table III describes that the user is editing the personal profile with little information provided and staying a short time on that page. Only based on this fact, there are many probabilities for the possible low- and high-level desires, so it is imperative for domain experts to consider available information comprehensively for elicitation.

E. Summary of Illustrative Example

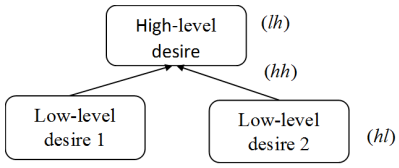

Fig. 6. NFRs presented between two different abstraction levels of desires (lh, hh, and hl map to the cases described in Table I)

Overall, our NFRs analysis based on desire inference confidence provides three promising ways to address the problem of discovering users' new NFRs. They can be mapped to cases presented in Table I and represented by a desire tree as described in Figure 6. For any consecutive two level of desires, case hh indicates that there is a new contributing link from an existing lower-level desire to an existing higher-level desire; case lh implies the existence of a new higher-level desire and the contributing relation from its corresponding subordinate low-level desires; case hl suggests that a new alternative that contributing to a higher-level desire is found. Domain experts can locate their interested situations by considering both their appearance frequency and desires inference confidence. Regarding the situation frequency, it is hard to find exact same situations, but it is possible to find situations satisfy a certain pattern. For example, in Table III, for case hh, the pattern could be any situation that has the same level- and high-level desires; for cases lh and hl, the pattern could be situations have a same low- and high- level desire, as well as actions or some extracted features based on system contexts, respectively. To look beyond the neighboring level desires, we suggest to reason from the lowest-level desires to the highest-level desires to explore the whole desire tree. The reason is that changes on the root (highest) level desire usually have greater impact on all the subordinate level desires, while changes on a bottom level desire may not have much influence on the root level desire or other branches of low-level desires.

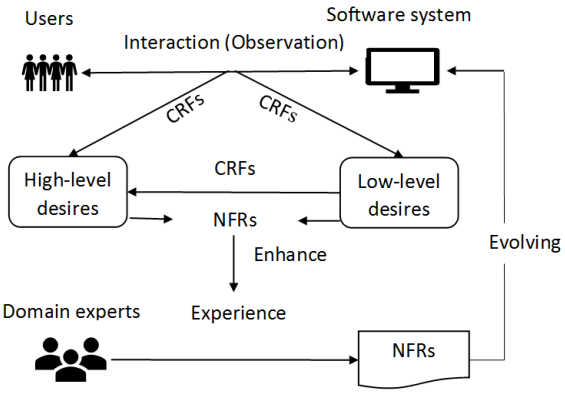

Fig. 7. The outline of multi-layered desires oriented NFRs analysis

Our NFRs analysis focuses on users' genuine desires under situation framework based on a statistical model CRFs. The outline of NFRs analysis with our method is presented in Figure 7. While users interacting with software system, their behaviors and system corresponding information will be provided as input data of CRF model to predict users' desires, at both high- and low-levels. Because desires are usually subjective, we try to achieve objectivity from observing users' behaviors, not through inferring desires. The relation of different abstraction level of desires could be reflected based on the analysis described in Section VI-D, such as new contributing relations (hh), high-level desires (lh), and alternatives (hl). After analyzing inferred desires and their time-stamped situations, the information of how a user want to address a desire will be revealed, which is a valuable resource to help domain experts to extract new NFRs for the system-to-be. The new NFRs provide information that users really need to help software system evolution.

VII. Discussion

Mining NFRs from requirement documentations. Most automatic NFRs extraction methods [8], [26], [38], [40] adopt natural language processing (NLP) techniques. They assume that different kinds of NFRs can be reflected by specific keywords, which serve as indicators. However, those methods mainly suffer from the low accuracy or high false positive rate. Although those techniques have advantages in speed, they cannot reflect the essential meaning of documentations. Our method to try to interpret users' mental states shows a promising direction to combine NLP based techniques and desires-oriented framework, such as speeding up the construction of initial knowledge base with automatically mined NFRs.

Agile Development. One of the important weaknesses of Agile Development when dealing with high-integrity software system is that it is lack of proven method in dealing with non-functional requirements [6]. Our method shows a promising opportunity to improve Agile Development from the requirements aspect. As the core characteristic of Agile Development is to change and iterate, the NFRs elicited in each iteration, as shown in Figure 4, can serve as alternative features to be considered in next development iteration. Also, our method requires a system/prototype available, and a prototype can usually be built quickly when Agile method is used. Thus, our method fits well to the purpose of iterative and incremental refinement of the software products in Agile development.

Alternative ways for desires inference. An alternative way to infer users' desires at a higher-level is to consider not only uses' observations data, but also its subordinate low-level desires. For example, In Table III, to predict the high-level desire in the last column, low-level desires can be taken into account. So the input data can be a combination of actions, context values, and low-level desires, or just the low-level desires. The criteria to decide what could be included in the combined input should be formulated empirically with regard to the practical, specific problem. The assumption to use low-level desires to infer high-level desires is based on the availability of low-level desires labeled with high confidences, which could be estimated by checking the prediction probability by CRFs models.

Hierarchical division of desires. Situation aware computing has been applied in requirements engineering for the purpose of finding users' emerging intentions by observing their behaviors and system contexts. The most relevant work to ours was conducted by Xie et al. [32]. They applied a statistical model CRFs to infer users' goal based on users' operation and system corresponding contexts. They also proposed three analysis methods to elicit users' emerging intentions by considering users' divergent behavior, goal transition, and erroneous behavior respectively. Comparing to their work, our approach have a few advantages: 1) we make further progress for true desires inference from merely goals inference to gain a better representation of human mental states; 2) we divide desires into multiple layers to analysis relations among desires from different layers, while goals are treated as a single layer concept in Xie's work; 3) as a basis for domain experts to refer and reason, we suggest a more concrete metric to utilize desires inference confidence in different layers for new NFRs elicitation.

Existing NFRs models One popular modeling method of NFRs is through soft-goal with i^* framework [35], which focuses on the early-phase requirement engineering (RE). Domain experts and software engineers build models to analyze stakeholders' interests and how they may be addressed. This approach tends to be subjective and labor-intensive since domain experts communicate with stakeholders in format of interviews, and it is hard to acquire the genuine requirements of the real users. Our method provides users' desires to domain experts after software systems have been deployed. Domain experts can take advantage of the data that indicate human mental states to calibrate the i^* model to represent more accurate alternatives and stakeholders' relationships.

VIII. Threats to Validity

The threats to validity are as follows:

Internal. To make CRFs model output reliable results, desires should be defined with an appropriate granularity. The too coarse-grained defined desires may cause the CRFs lose the ability to generalize the relations between contextual data and user desires. The too fine-grained described desires will make the CRFs confused by given excess labels to the model. Thus, in practice, domain experts shall define the desires with distinction and consider the granularity as well. Additionally, the emerging desires for NFRs elicitation are mainly reasoned by domain experts based on subjective judgment. Thus, we recommend including statistical tests to determine when the domain experts have reached consensus.

Domain. We utilized the CoRE system as an example to illustrate our NFRs elicitation method. We believe that our method is generally applicable to other human-centric context-aware domains because the training process will not be hard to transfer across systems as discussed in SectionV-A.

External. We have not run experiment with human subjects. We will conduct experiment to validate our method as our IRB application has just been approved.

IX. Future Work

Our multi-layered desires based NFRs elicitation method should be validated, at the first step, if it can infer human desires accurately. We are in the progress of running an empirical study to verify our assumptions. The experimental system for the coming study has been designed, and we plan to recruit students and faculties in a few universities. Our IRB (Institutional Review Board) application has just been approved, and we will start collecting data to validate our method.

X. Conclusion

This paper proposes a method to analyze and elicit NFRs from human desires based on a situation framework. We differentiate desires with different abstraction levels and treat

NFRs as how users realize a higher-level desire through lower-level desires. Our main contribution is to analysis NFRs from human-centric dimension, which, we believe, is a good fit and closely corresponds to the substantial characteristics of NFRs. We also take inference confidence into account, to make better sense of CRFs inference results. We reason through an illustrative example and show that our method can be promising in NFRs elicitation, and much work remains to be done on this foundation.

REFERENCES

[1] 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.

[2] D. Ameller, X. Franch, and J. Cabot. Dealing with non-functional requirements in model-driven development. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 189–198. IEEE, 2010.

[3] 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.

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

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

[12] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1):32–64, 1995.

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

[14] 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.

[15] 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.

[16] 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.

[17] A. Knauss, D. Damian, and K. Schneider. Eliciting contextual requirements at design time: A case study. In *Empirical Requirements Engineering (EmpiRE), 2014 IEEE Fourth International Workshop on*, pages 56–63. IEEE, 2014.

[18] Z. Kurtanovic and W. Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning.

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

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] B. Wei, B. Yin, Z. Jin, and D. Zowghi. rσ: Automated reasoning tool for non-functional requirement goal models. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 337–338. IEEE, 2011.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang. An empirical study on stack overflow using topic analysis. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 446–449. IEEE Press, 2015.